GRE 및 IPsec을 사용하여 IPv4 단편화, MTU, MSS 및 PMTUD 문제를 해결합니다.

목차

소개

이 문서에서는 IPv4 프래그먼트화와 PMTUD(Path Maximum Transmission Unit Discovery)의 작동 방식을 설명합니다.

배경 정보

또한 다양한 IPv4 터널 조합과 함께 PMTUD의 동작과 관련된 시나리오를 설명합니다.

IPv4 프래그먼트화 및 재조합

IPv4 데이터그램의 최대 길이는 65535이지만, 대부분의 전송 링크는 이보다 더 작은 최대 패킷 길이 제한(MTU 라고 함)을 적용합니다. MTU 값은 전송 링크의 유형에 따라 다릅니다.

설계상 IPv4는 필요에 따라 라우터에 IPv4 데이터그램을 프래그먼트화하도록 허용하기 때문에 MTU 차이를 수용합니다.

수신 스테이션이 프래그먼트를 원래의 전체 크기 IPv4 데이터그램으로 재조합합니다.

IPv4 프래그먼트화는 데이터그램을 나중에 재조합할 수 있는 여러 부분으로 나누는 작업입니다.

IPv4 프래그먼트화와 재조합에는 IPv4 헤더의 MF("more fragments") 플래그와 DF("don't fragment") 플래그와 함께 IPv4 source, destination, identification, total length 및 fragment offset 필드가 사용됩니다.

IPv4 프래그먼트화와 재조합의 메커니즘에 대한 자세한 내용은 RFC 791을 참조하십시오.

아래 이미지는 IPv4 헤더의 레이아웃을 나타낸 것입니다.

Original IP Datagram

Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0	345	5140	0	0	0

IP Fragments (Ethernet)

Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0-0	345	1500	0	1	0
0-1	345	1500	0	1	185
0-2	345	1500	0	1	370
0-3	345	700	0	0	555

identification은 16비트이며, IPv4 데이터그램의 발신 장치가 할당한 값입니다. 이는 데이터그램의 프래그먼트 재조합에 도움이 됩니다.

fragment offset은 13비트이며, 원래 IPv4 데이터그램에서 프래그먼트가 속한 위치를 나타냅니다. 이 값은 8바이트의 배수입니다.

IPv4 헤더의 flags 필드에는 제어 플래그에 대한 3개 비트가 있습니다. DF("do not fragment") 비트는 패킷의 프래그먼트화 허용 여부를 결정합니다.

Bit 0은 예약되며 항상 0으로 설정됩니다.

Bit 1은 DF 비트(0 = "can fragment", 1 = "do not fragment")입니다.

Bit 2는 MF 비트(0 = "last fragment," 1 = "more fragments")입니다.

가치	Bit 0 예약됨	Bit 1 DF	Bit 2 MF
0	0	5월	Last
1	0	Do not	기타

IPv4 프래그먼트의 길이를 모두 합하면 값이 원래 IPv4 데이터그램 길이를 60만큼 초과합니다.

총 길이가 60만큼 늘어난 이유는 IPv4 헤더가 3개 추가로 생성되었기 때문입니다(첫 번째 프래그먼트 다음에 각 프래그먼트에 대해 하나씩).

첫 번째 프래그먼트의 오프셋은 0이고, 이 프래그먼트의 길이는 1500입니다. 이는 약간 수정된 원래 IPv4 헤더의 20바이트가 포함된 길이입니다.

두 번째 프래그먼트의 오프셋은 185(185 x 8 = 1480)이고, 이 프래그먼트의 데이터 부분이 원래 IPv4 데이터그램의 1480바이트부터 시작합니다.

이 프래그먼트의 길이는 1500으로, 이 프래그먼트에 대해 생성된 추가 IPv4 헤더가 포함된 길이입니다.

세 번째 프래그먼트의 오프셋은 370(370 x 8 = 2960)이고, 이 프래그먼트의 데이터 부분이 원래 IPv4 데이터그램의 2960바이트부터 시작합니다.

이 프래그먼트의 길이는 1500으로, 이 프래그먼트에 대해 생성된 추가 IPv4 헤더가 포함된 길이입니다.

네 번째 프래그먼트의 오프셋은 555(555 x 8 = 4440)이고, 이 프래그먼트의 데이터 부분이 원래 IPv4 데이터그램의 4440바이트부터 시작함을 의미합니다.

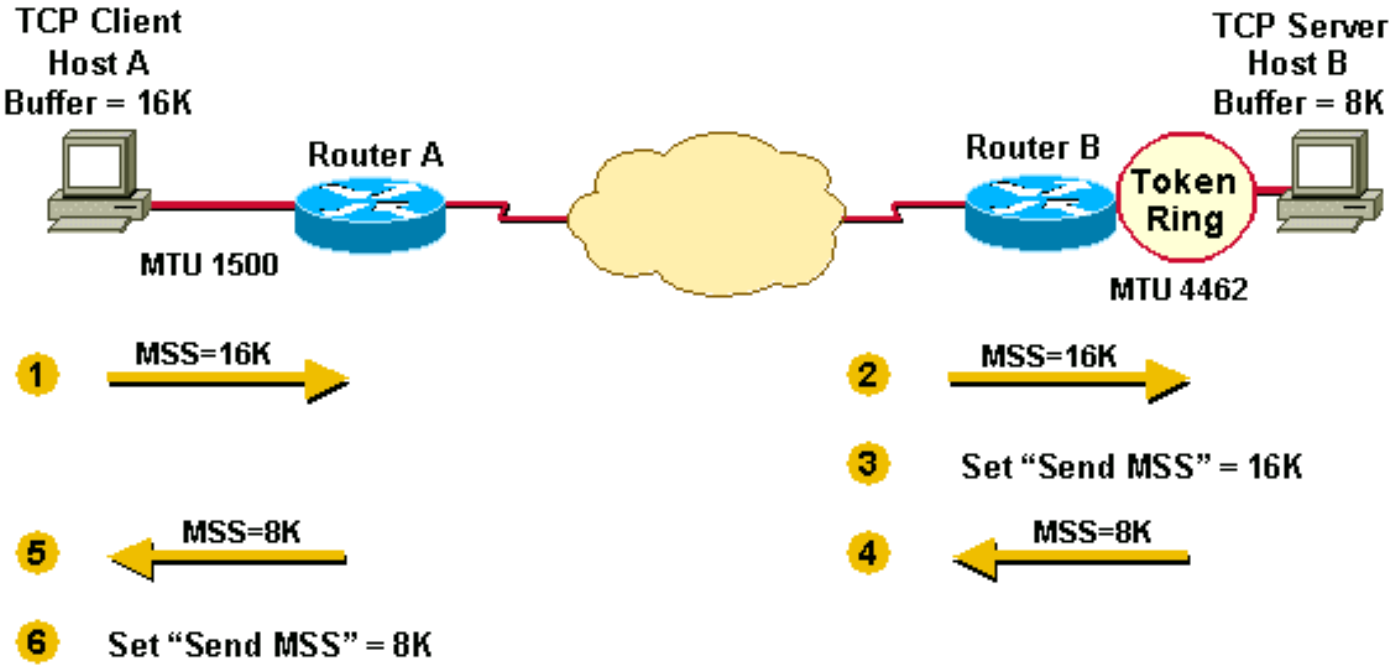
이 프래그먼트의 길이는 700바이트로, 이 프래그먼트에 대해 생성된 추가 IPv4 헤더가 포함된 길이입니다.

마지막 프래그먼트가 수신된 경우에만 원래 IPv4 데이터그램의 크기를 확인할 수 있습니다.

마지막 프래그먼트의 프래그먼트 오프셋(555)으로 원래 IPv4 데이터그램의 데이터 오프셋이 4440바이트가 됩니다.

마지막 프래그먼트에서 데이터 바이트(680 = 700-20)를 더하면 원래 IPv4 데이터그램의 데이터 부분인 5120바이트가 됩니다.

IPv4 헤더에 대한 20바이트를 더하면 그림에 표시된 것처럼 원래 IPv4 데이터그램의 크기(4440 + 680 + 20 = 5140)와 같습니다.



IPv4 프래그먼트화 문제

IPv4 프래그먼트화로 인해 IPv4 데이터그램 프래그먼트화를 위한 CPU 및 메모리 오버헤드가 조금 증가하게 됩니다. 이는 발신 장치와 수신 장치 간의 경로에 있는 라우터에도 해당되고 발신 장치에도 해당됩니다.

프래그먼트 생성에는 프래그먼트 헤더를 생성하고 원래 데이터그램을 프래그먼트에 복사하는 작업이 포함됩니다.

이 작업은 프래그먼트를 생성하는 데 필요한 정보를 즉시 사용할 수 있기 때문에 효율적으로 이루어집니다.

프래그먼트화는 프래그먼트를 재조합할 때 수신 장치에 더 많은 오버헤드를 초래합니다. 수신 장치가 도착한 프래그먼트에 메모리를 할당하고, 모든 프래그먼트가 도착하면 프래그먼트를 하나의 데이터그램으로 다시 합해야 하기 때문입니다.

호스트에서의 재조합은 문제가 되지 않는데, 그 이유는 호스트에 이 작업에 투입할 시간과 메모리 리소스가 있기 때문입니다.

하지만 최대한 빨리 패킷을 전달하는 것이 기본 작업인 라우터에서는 재조합이 비효율적입니다.

라우터는 일정 시간 동안 패킷을 가지고 있도록 설계되지 않습니다.

재조합을 수행하는 라우터는 사용할 수 있는 버퍼 중 가장 큰 버퍼(18K)를 선택합니다. 마지막 프래그먼트가 도착할 때까지 원래 IPv4 패킷의 크기를 파악할 방법이 없기 때문입니다.

또 다른 프래그먼트화 문제는 삭제된 프래그먼트가 처리되는 방식과 관련 있습니다.

IPv4 데이터그램의 한 프래그먼트가 삭제되면 원래 IPv4 데이터그램 전체가 있어야 하며, 역시 프래그먼트화됩니다.

이는 NFS(Network File System)에서 볼 수 있습니다. NFS의 읽기 및 쓰기 블록 크기는 8192입니다

.

따라서 NFS IPv4/UDP 데이터그램이 약 8500바이트(NFS, UDP 및 IPv4 헤더 포함)입니다.

이더넷(MTU 1500)에 연결된 발신 스테이션은 8500바이트 데이터그램을 6개로 프래그먼트화해야 합니다. 5개는 1500바이트 프래그먼트이고 1개는 1100바이트 프래그먼트입니다.

링크가 혼잡하여 6개 프래그먼트 중 하나라도 삭제되면 완전한 원래 데이터그램을 재전송해야 합니다. 즉, 프래그먼트를 6개 더 생성해야 합니다.

이 링크가 6개 패킷 중 하나를 삭제하는 경우, NFS 데이터가 이 링크를 통해 전송될 가능성이 낮습니다. 각 NFS 8500바이트의 원래 IPv4 데이터그램에서 하나 이상의 IPv4 프래그먼트가 삭제되기 때문입니다.

L4(Layer 4)를 기반으로 L7(Layer 7) 정보를 통해 패킷을 필터링하거나 조작하는 방화벽은 IPv4 프래그먼트를 올바르게 처리하는 데 어려움을 겪을 수 있습니다.

IPv4 프래그먼트가 제대로 설정되지 않은 경우 패킷 필터와 일치하는 정보를 갖고 있지 않기 때문에, 방화벽이 비 초기 프래그먼트를 차단합니다.

따라서 원래 IPv4 데이터그램을 수신 호스트에서 재조합하지 못할 수 있습니다.

정보가 불충분한 비 초기 프래그먼트를 필터와 적절히 일치시키는 것을 허용하도록 방화벽이 설정되면, 방화벽을 통한 비 초기 프래그먼트 공격이 발생할 수 있습니다.

Content Switch Engine과 같은 네트워크 디바이스의 경우 L4~L7 정보를 기반으로 패킷을 전달하며, 패킷이 여러 프래그먼트로 나뉘면 네트워크 디바이스가 정책을 적용하는 데 어려움을 겪습니다.

IPv4 프래그먼트화 방지: TCP MSS의 작동 방식

TCP(Transmission Control Protocol) MSS(Maximum Segment Size)는 호스트가 수락하는 단일 TCP/IPv4 데이터그램의 최대 데이터 크기를 정의합니다.

이 TCP/IPv4 데이터그램은 IPv4 레이어에서 프래그먼트화될 수 있습니다. MSS 값은 TCP SYN 세그먼트에서만 TCP 헤더 옵션으로 전송됩니다.

TCP 연결의 한 쪽에서 다른 쪽으로 해당 MSS 값을 보고합니다. MSS 값은 호스트 간에 협상되지 않습니다.

단일 TCP 세그먼트에 있는 데이터 크기를 수신 호스트에서 보고한 MSS 값 이하로 제한하기 위해 전송 호스트가 필요합니다.

원래, MSS는 단일 IPv4 데이터그램에 포함된 TCP 데이터를 저장할 수 있도록 수신 스테이션에 할당된 버퍼 크기(65496바이트보다 크거나 같음)였습니다.

그리고 TCP 수신 장치가 수락하는 데이터의 최대 세그먼트 크기였습니다. 이 TCP 세그먼트는 크기가 최대 64000일 수 있으며, 수신 호스트로 전송하기 위해 IPv4 레이어에서 프래그먼트화될 수 있습니다.

수신 호스트는 완전한 TCP 세그먼트를 TCP 레이어에 전달하기 전에 IPv4 데이터그램을 재조합합

니다.

TCP 세그먼트 및 IPv4 데이터그램 크기를 제한하기 위해 MSS 값이 설정 및 사용되는 방식.

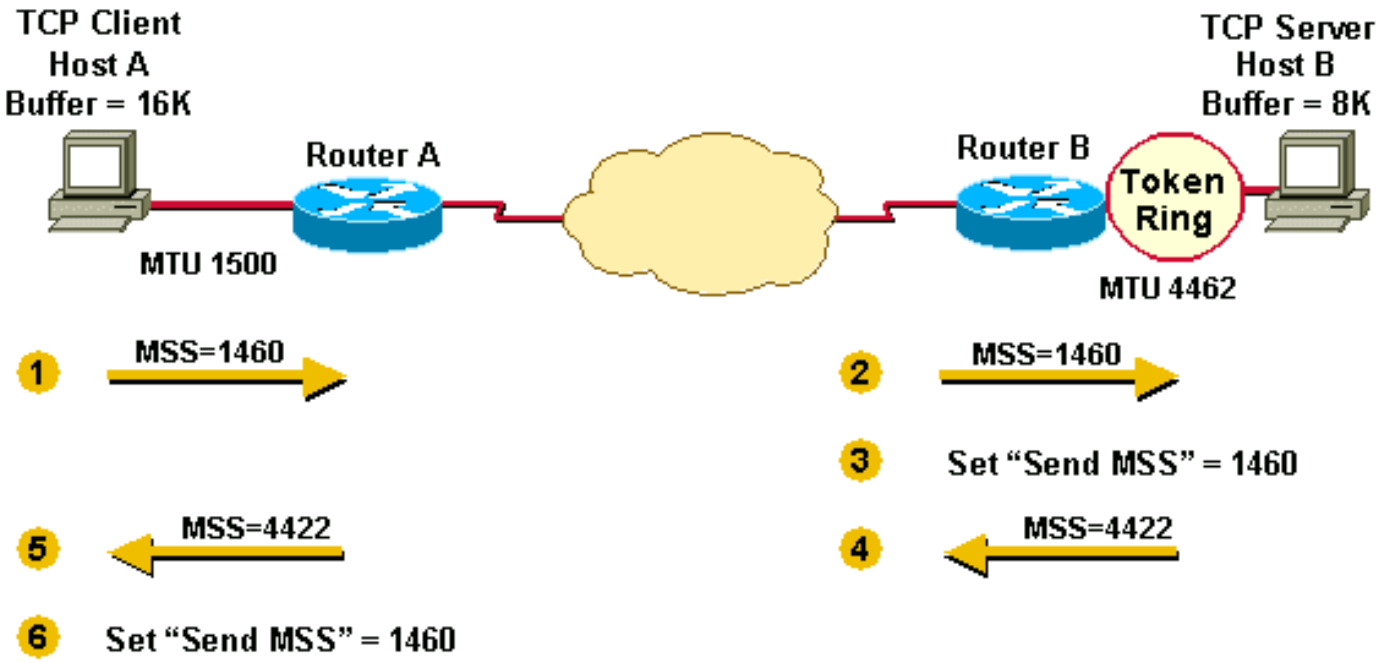
예시 1은 MSS가 처음에 어떻게 구현되었는지를 보여줍니다.

호스트 A의 버퍼는 16K이고 호스트 B의 버퍼는 8K입니다. 이 두 호스트는 MSS 값을 주고받고, 서로에게 데이터를 전송하기 위해 전송 MSS를 조정합니다.

TCP 스택이 16000바이트 또는 8000바이트의 데이터를 IPv4에 전달하기 때문에, 호스트 A와 호스트 B는 인터페이스 MTU보다 크지만 전송 MSS보다 작은 IPv4 데이터그램을 프래그먼트화해야 합니다.

호스트 B의 경우, 패킷이 토큰 링 LAN에 도달하기 위해, 그리고 다시 이더넷 LAN에 도달하기 위해 프래그먼트화됩니다.

예 1



1. 호스트 A는 MSS 값 16K를 호스트 B에 전송합니다.
2. 호스트 B는 호스트 A에서 16K MSS 값을 받습니다.
3. 호스트 B는 전송 MSS 값을 16K로 설정합니다.
4. 호스트 B는 MSS 값 8K를 호스트 A에 전송합니다.
5. 호스트 A는 호스트 B에서 8K MSS 값을 받습니다.
6. 호스트 A는 전송 MSS 값을 8K로 설정합니다.

TCP 연결의 엔드포인트에서 IPv4 프래그먼트화가 발생하는 것을 방지하기 위해, 선택한 MSS 값이 발신 인터페이스의 MTU(- 40)와 최소 버퍼 크기로 변경되었습니다.

MSS 값은 MTU 값보다 40바이트 작습니다. MSS가 20바이트의 IPv4 헤더와 20바이트의 TCP 헤더가 포함되지 않은 단순한 TCP 데이터 크기이기 때문입니다.

MSS는 기본 헤더 크기를 기반으로 합니다. 발신 장치 스택은 사용된 TCP 옵션 또는 IPv4 옵션에 따라 IPv4 헤더의 값 또는 TCP 헤더의 값을 빼야 합니다.

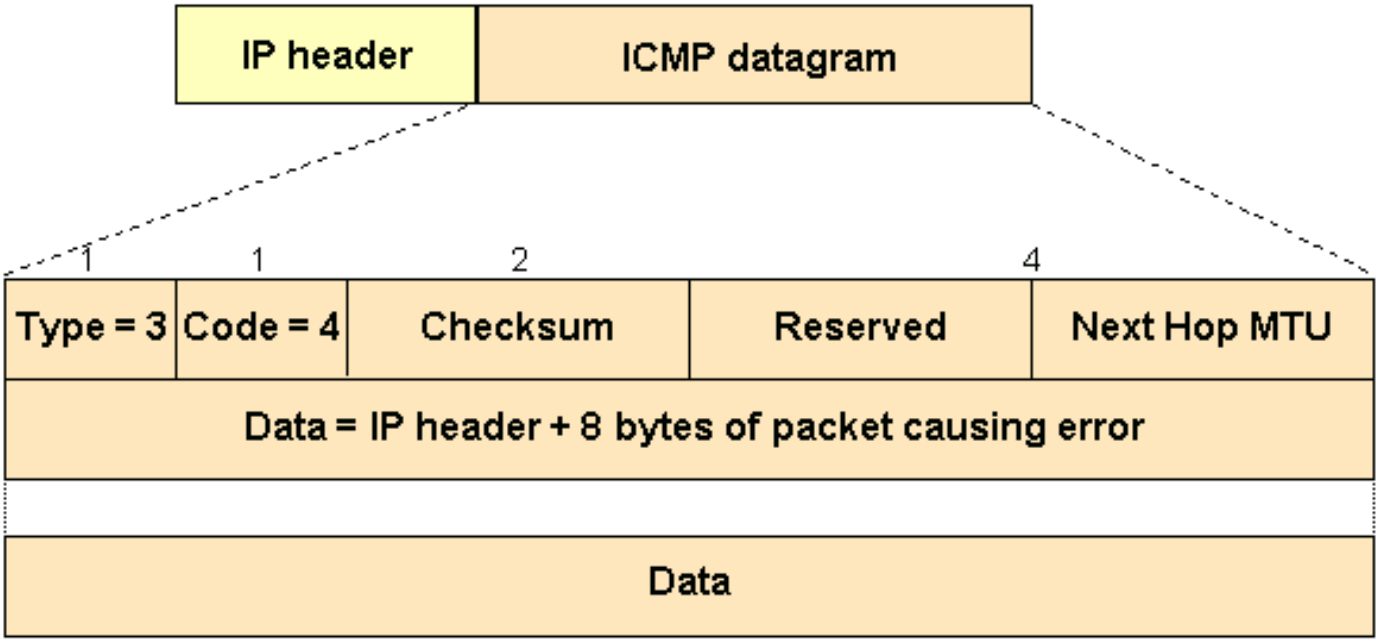
MSS는 현재 각 호스트가 먼저 발신 인터페이스 MTU를 자체 버퍼와 비교한 다음, 더 낮은 값을 전송 MSS로 선택하는 방식으로 작동합니다.

그러면 호스트가 수신된 MSS 크기를 자체 인터페이스 MTU와 비교하고, 두 가지 값 중 더 낮은 값을 다시 선택합니다.

예시 2에서는 로컬 회선과 원격 회선에서의 프래그먼트화를 방지하기 위해 발신 장치가 수행하는 추가 단계를 보여줍니다.

호스트가 각자의 MSS 값을 서로 전송하기 전에 각 호스트에서 발신 인터페이스의 MTU가 고려됩니다. 이는 프래그먼트화 방지에 도움이 됩니다.

예 2



1. 호스트 A가 MSS 버퍼(16K)와 MTU(1500 - 40 = 1460)를 비교하고, 더 낮은 값을 MSS(1460)로 사용하여 호스트 B에 전송합니다.
2. 호스트 B는 호스트 A의 전송 MSS(1460)를 수신하고 이 값을 아웃바운드 인터페이스의 값 MTU - 40(4422)과 비교합니다.
3. 호스트 B가 IPv4 데이터그램을 호스트 A에 전송하기 위해 더 낮은 값(1460)을 MSS로 설정합니다.
4. 호스트 B가 MSS 버퍼(8K)와 MTU(4462-40 = 4422)를 비교하고, 4422를 MSS로 사용하여 호스트 A에 전송합니다.
5. 호스트 A는 호스트 B의 전송 MSS(4422)를 수신하고 이 값을 아웃바운드 인터페이스의 값 MTU - 40(1460)과 비교합니다.
6. 호스트 A는 IPv4 데이터그램을 호스트 B에 전송하기 위해 더 낮은 값(1460)을 MSS로 설정합니다.

1460이 두 호스트가 서로를 위해 선택한 전송 MSS 값입니다. TCP 연결의 각 끝에서 전송 MSS 값

은 대체로 동일합니다.

예시 2에서는 양쪽의 발신 인터페이스 MTU가 호스트에서 고려되기 때문에 TCP 연결의 엔드포인트에서 프래그먼트화가 발생하지 않습니다.

패킷이 양쪽 호스트의 아웃바운드 인터페이스보다 MTU가 낮은 링크를 발견하는 경우에는 라우터 A와 라우터 B 사이의 네트워크에서 패킷이 여전히 프래그먼트화됩니다.

PMTUD는 무엇입니까

TCP MSS는 TCP 연결의 두 엔드포인트에서 발생하는 프래그먼트화를 처리하지만, 이러한 두 엔드포인트 중간에 더 작은 MTU 링크가 있는 경우는 처리하지 않습니다.

PMTUD는 엔드포인트 간의 경로에서 프래그먼트화가 발생하는 것을 방지하기 위해 개발되었습니다. PMTUD는 패킷의 소스에서 대상까지의 경로에서 가장 낮은 MTU를 동적으로 확인하는 데 사용됩니다.

참고: PMTUD는 TCP와 UDP에서만 지원됩니다. 다른 프로토콜에서는 지원되지 않습니다. PMTUD가 호스트에서 활성화되어 있는 경우 호스트의 모든 TCP 패킷과 UDP 패킷에 DF 비트가 설정됩니다.

DF 비트가 설정된 전체 MSS 데이터 패킷을 호스트가 전송할 경우, PMTUD는 패킷에 프래그먼트화가 필요하다는 정보를 수신하면 연결을 위해 전송 MSS 값을 줄입니다.

호스트는 라우팅 테이블에 host(/32) 항목을 MTU 값과 함께 생성하므로 대상에 대한 이 MTU 값을 기록합니다.

라우터가 DF 비트가 설정된 IPv4 데이터그램을 패킷 크기보다 MTU가 낮은 링크로 전달하려고 하면 라우터는 패킷을 삭제합니다. 그런 다음 "fragmentation needed and DF set"(type 3, code 4) 코드가 포함된 ICMP(Internet Control Message Protocol) "Destination Unreachable" 메시지를 IPv4 데이터그램 소스에 반환합니다.

소스 스테이션이 ICMP 메시지를 수신하면 전송 MSS를 줄이고, TCP가 세그먼트를 재전송할 때 더 작아진 세그먼트 크기를 사용합니다.

다음은 명령이 켜진 후 라우터에 표시되는 ICMP "fragmentation needed and DF set" 메시지의 **debug ip icmp** 예입니다.

```
ICMP: dst (10.10.10.10) frag. needed and DF set
unreachable sent to 10.1.1.1
```

아래 다이어그램에는 "fragmentation needed and DF set" "Destination Unreachable" 메시지의 ICMP 헤더 형식이 나와 있습니다.

Plateau	MTU	Comments	Reference
	65535	Official maximum MTU	RFC 791
	65535	Hyperchannel	RFC 1044
65535			
32000		Just in case	
	17914	16Mb IBM Token Ring	ref. [6]
17914			
	8166	IEEE 802.4	RFC 1042
8166			
	4464	IEEE 802.5 (4Mb max)	RFC 1042
	4352	FDDI (Revised)	RFC 1188
4352 (1%)			
	2048	Wideband Network	RFC 907
	2002	IEEE 802.5 (4Mb recommended)	RFC 1042
2002 (2%)			
	1536	Exp. Ethernet Nets	RFC 895
	1500	Ethernet Networks	RFC 894
	1500	Point-to-Point (default)	RFC 1134
	1492	IEEE 802.3	RFC 1042
1492 (3%)			
	1006	SLIP	RFC 1055
	1006	ARPANET	BBN 1822
1006			
	576	X.25 Networks	RFC 877
	544	DEC IP Portal	ref. [10]
	512	NETBIOS	RFC 1088
	508	IEEE 802/Source-Rt Bridge	RFC 1042
	508	ARCNET	RFC 1051
508 (13%)			
	296	Point-to-Point (low delay)	RFC 1144
296			
68		Official minimum MTU	RFC 791

RFC 1191에 따라 "fragmentation needed and DF set"을 나타내는 ICMP 메시지를 반환하는 라우터는 하위 16비트의 ICMP 추가 헤더 필드(ICMP 사양 RFC 792에서 "unused"로 레이블이 지정됨)에 다음 홉 네트워크의 MTU를 포함해야 합니다.

RFC 1191의 초기 구현에서는 다음 홉의 MTU 정보를 제공하지 않았습니다. 이 정보가 제공되었더라도 일부 호스트는 이 정보를 무시합니다.

이러한 경우, RFC 1191에는 PMTUD 중에 MTU의 감소 폭으로 제안된 값의 테이블도 포함됩니다.

이 예시에 표시된 대로 이 테이블은 호스트가 합리적인 전송 MSS 값을 더 빨리 찾는 데 사용됩니다
.

발신 장치와 수신 장치 간의 경로가 동적으로 변경될 수 있으므로 PMTUD는 모든 패킷에 대해 지속적으로 실시됩니다.

발신 장치는 "Cannot Fragment" ICMP 메시지를 수신할 때마다 라우팅 정보(여기에 PMTUD 저장)를 업데이트합니다.

PMTUD 중에는 다음과 같은 두 가지 경우가 발생할 수 있습니다.

1. 패킷이 프래그먼트화되지 않은 상태로 수신 장치까지 갈 수 있습니다.

참고: 라우터는 DoS 공격으로부터 CPU를 보호하기 위해 보내는 ICMP 연결 불가 메시지 수를 초당 2개로 제한합니다. 따라서 이 컨텍스트에서 라우터가 초당 2개 이상의 ICMP 메시지(유형 = 3, 코드 = 4)로 응답해야 하는 네트워크 시나리오가 있는 경우(호스트가 다를 수 있음) 명령을 사용하여 ICMP 메시지 제한을 **no ip icmp rate-limit unreachable [df] interface** 비활성화합니다.

---

2. 발신 장치는 수신 장치의 경로에 있는 홉에서 ICMP "Cannot Fragment" 메시지를 받을 수 있습니다.

PMTUD는 TCP 플로우의 한쪽 방향마다 독립적으로 수행됩니다.

따라서 플로우의 한쪽 방향에서 PMTUD가 종단 스테이션 중 하나에서 전송 MSS를 낮추도록 하지만, 다른 종단 스테이션은 PMTUD를 유발할 정도로 큰 IPv4 데이터그램을 전송하지 않았기 때문에 원래 전송 MSS를 유지하는 경우가 있습니다.

그 예시는 예시 3에 나와 있는 HTTP 연결입니다. TCP 클라이언트는 작은 패킷을 보내고 서버는 큰 패킷을 보냅니다.

이 경우 서버의 대형 패킷(576바이트 초과)만 PMTUD를 유발합니다.

클라이언트의 패킷은 크기가 작아(576바이트 미만), 576 MTU 링크를 지나는 데 프래그먼트화가 필요하지 않기 때문에 PMTUD를 유발하지 않습니다.

예 3

예시 4에서는 경로 중 하나가 다른 경로보다 더 작은 최소 MTU를 갖는 비대칭 라우팅의 예시를 보여줍니다.

비대칭 라우팅은 두 엔드포인트 간에 데이터를 주고받을 때 서로 다른 경로를 사용하는 경우에 발

생합니다.

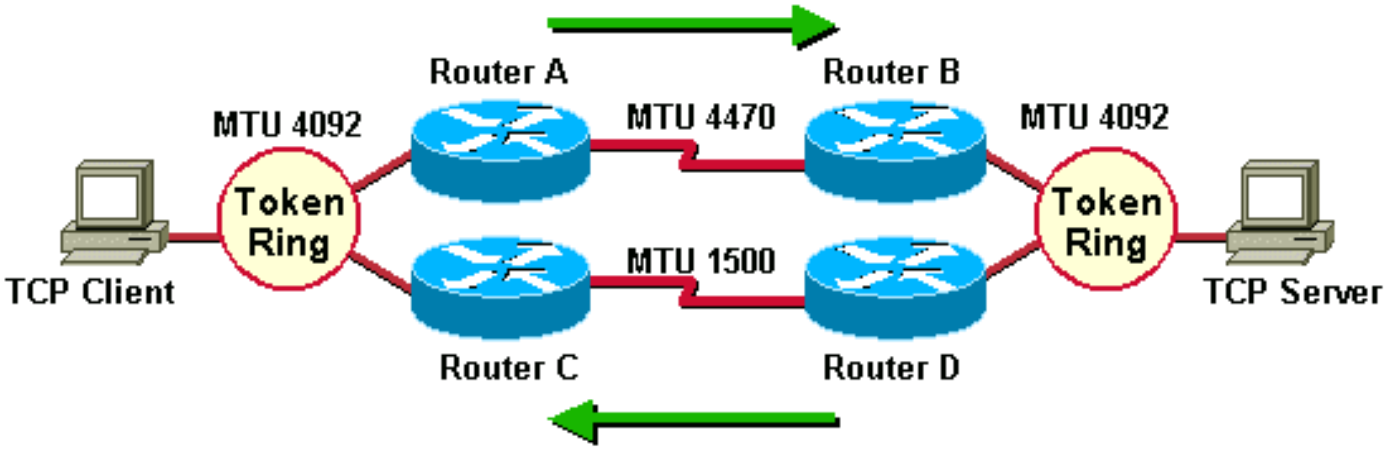
이 예시에서 PMTUD는 TCP 플로우의 한쪽 방향에서만 전송 MSS를 낮추도록 합니다.

TCP 클라이언트에서 서버로 향하는 트래픽은 라우터 A와 라우터 B를 거치는 반면, 서버에서 클라이언트로 향하는 반환 트래픽은 라우터 D와 라우터 C를 거칩니다.

TCP 서버가 클라이언트에 패킷을 전송하면, PMTUD는 서버에서 전송 MSS를 낮추도록 합니다. 라우터 D가 라우터 C에 4092바이트 패킷을 전송하려면 먼저 이 패킷을 프래그먼트화해야 하기 때문입니다.

반면 클라이언트는 "fragmentation needed and DF set"을 나타내는 코드가 있는 ICMP "Destination Unreachable" 메시지를 수신하지 않습니다. 라우터 A가 라우터 B를 통해 서버에 패킷을 전송할 때 패킷을 프래그먼트화할 필요가 없기 때문입니다.

예 4



참고: ip tcp path-mtu-discovery 명령은 라우터(예: BGP 및 텔넷)에 의해 시작된 TCP 연결에 대한 TCP MTU 경로 검색을 활성화하는 데 사용됩니다.

PMTUD 문제

이로 인해 PMTUD가 중단될 수 있습니다.

- 라우터가 패킷을 삭제하고 ICMP 메시지를 보내지 않습니다. (일반적이지 않음)
- 라우터가 ICMP 메시지를 생성하고 전송하지만, ICMP 메시지가 이 라우터와 발신 장치 사이에 있는 라우터 또는 방화벽에 의해 차단됩니다. (일반적임)
- 라우터가 ICMP 메시지를 생성하고 전송하지만, 발신 장치가 메시지를 무시합니다. (일반적이지 않음)

세 가지 항목 중 첫 번째와 마지막은 대체로 오류의 결과이지만, 두 번째 항목은 일반적인 문제입니다.

ICMP 패킷 필터를 구현하는 경우 특정 ICMP 메시지 유형만 차단하는 것이 아니라 모든 ICMP 메시지 유형을 차단하려는 경향이 있습니다.

패킷 필터는 "unreachable" 또는 "time-exceeded" 메시지를 제외한 모든 ICMP 메시지 유형을 차단할 수 있습니다.

PMTUD의 성공 또는 실패는 ICMP 연결 불가 메시지가 TCP/IPv4 패킷의 발신 장치에 도달하는지 여부에 따라 결정됩니다.

ICMP time-exceeded 메시지는 다른 IPv4 문제에 중요합니다.

다음은 이러한 패킷 필터의 예로, 라우터에 구현된 패킷 필터입니다.

```
access-list 101 permit icmp any any unreachable
access-list 101 permit icmp any any time-exceeded
access-list 101 deny icmp any any
access-list 101 permit ip any any
```

ICMP가 완전히 차단되는 문제를 완화하는 데 사용할 수 있는 다른 방법이 있습니다.

- 라우터에서 DF 비트를 지우고 프래그먼트화를 허용하는 것입니다. 하지만 이는 좋은 방법이 아닙니다. 자세한 내용은 IP 프래그먼트화 문제를 참조하십시오.
- interface 명령으로 TCP MSS 옵션 값 MSS를 **ip tcp adjust-mss <500-1460>**조작합니다.

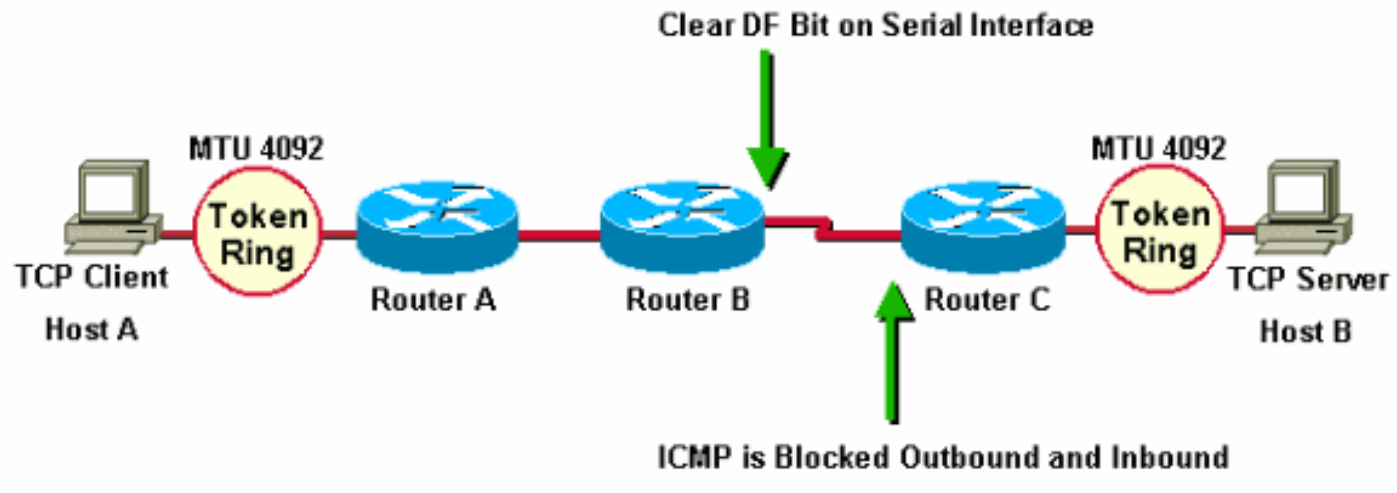
다음 예시에서는 라우터 A와 라우터 B가 동일한 관리 도메인에 있습니다. 라우터 C가 액세스 불가능하고 ICMP를 차단하므로 PMTUD가 중단됩니다.

이 상황을 해결하는 방법은 프래그먼트화를 허용하도록 라우터 B에서 양방향으로 DF 비트를 지우는 것입니다. 이 작업은 정책 라우팅을 사용하여 수행할 수 있습니다.

DF 비트를 지우는 구문은 Cisco IOS® Software Release 12.1(6) 이상에서 사용할 수 있습니다.

```
interface serial0
...
ip policy route-map clear-df-bit
route-map clear-df-bit permit 10
    match ip address 111
    set ip df 0

access-list 111 permit tcp any any
```



또 다른 옵션은 라우터를 통과하는 SYN 패킷의 TCP MSS 옵션 값을 변경하는 것입니다(Cisco IOS® 12.2(4) T 이상에서 사용 가능).

이렇게 하면 TCP SYN 패킷의 MSS 옵션 값이 줄어들어 명령의 값(1460)보다 **ip tcp adjust-mss**작아집니다.

그에 따라 TCP 발신 장치가 이 값 이하의 세그먼트를 전송하게 됩니다.

IPv4 패킷 크기는 TCP 헤더(20바이트)와 IPv4 헤더(20바이트)를 나타내기 위해 MSS 값(1460바이트)보다 40바이트 더 큽니다(1500).

이 명령을 사용하여 TCP SYN 패킷의 MSS를 조정할 수 **ip tcp adjust-mss** 있습니다. 이 구문은 TCP 세그먼트의 MSS 값을 1460으로 줄입니다.

이 명령은 인터페이스 serial0의 인바운드와 아웃바운드 트래픽 양쪽에 영향을 미칩니다.

```
int s0
ip tcp adjust-mss 1460
```

IPv4 터널이 더욱 광범위하게 구축되고 있기 때문에 IPv4 프래그먼트화 문제가 확산되고 있습니다.

터널이 프래그먼트화를 더 많이 유발하는데, 터널 캡슐화가 패킷 크기에 "오버헤드"를 추가하기 때문입니다.

예를 들어, GRE(Generic Router Encapsulation)가 추가되면 패킷에 24바이트가 추가되고, 이 크기 증가로 패킷이 아웃바운드 MTU보다 커지게 되므로 패킷 프래그먼트화가 필요합니다.

PMTUD가 필요한 일반 네트워크 토폴로지

PMTUD는 중간 링크의 MTU가 끝 링크의 MTU보다 작은 네트워크 상황에서 필요합니다. 이렇게 더 작은 MTU 링크가 존재하는 데에는 다음과 같은 몇 가지 일반적인 이유가 있습니다.

- 토큰 링(또는 FDDI)이 연결된 종단 호스트(이더넷 연결로 서로 연결됨). 끝에 있는 토큰 링(또는 FDDI)이 중간에 있는 이더넷 MTU보다 큽니다.
- PPPoE(ADSL과 함께 자주 사용됨)가 헤더에 사용할 8바이트를 필요로 합니다. 이 때문에 이더넷의 유효 MTU가 1492(1500-8)로 감소됩니다.

GRE, IPv4sec, L2TP 같은 터널 프로토콜도 각자의 헤더와 트레일러를 위한 공간을 필요로 합니다. 이 경우에도 아웃바운드 인터페이스의 유효 MTU가 감소됩니다.

터널

터널은 승객 패킷을 전송 프로토콜 내부에서 캡슐화하는 방법을 제공하는 Cisco 라우터의 논리적 인터페이스입니다.

point-to-point 캡슐화 체계를 구현하기 위한 서비스를 제공하도록 설계된 아키텍처입니다. 터널 인터페이스는 다음과 같은 3가지 기본 요소로 구성됩니다.

- 승객 프로토콜(AppleTalk, Banyan VINES, CLNS, DECnet, IPv4 또는 IPX)
- 캐리어 프로토콜 - 다음 캡슐화 프로토콜 중 하나임.
    - GRE - Cisco 멀티프로토콜 캐리어 프로토콜. 자세한 내용은 RFC 2784 및 RFC 1701을 참조하십시오.
    - IPv4 터널의 IPv4 - 자세한 내용은 RFC 2003을 참조하십시오.
- 전송 프로토콜 - 캡슐화된 프로토콜을 전달하는 데 사용되는 프로토콜입니다.

이 섹션에 표시된 패킷은 IPv4 터널링 개념을 설명합니다. 이때 GRE가 캡슐화 프로토콜이고 IPv4가 전송 프로토콜입니다.

승객 프로토콜도 IPv4입니다. 이 경우, IPv4는 전송 프로토콜이자 승객 프로토콜입니다.

일반 패킷

IPv4	TCP	Telnet

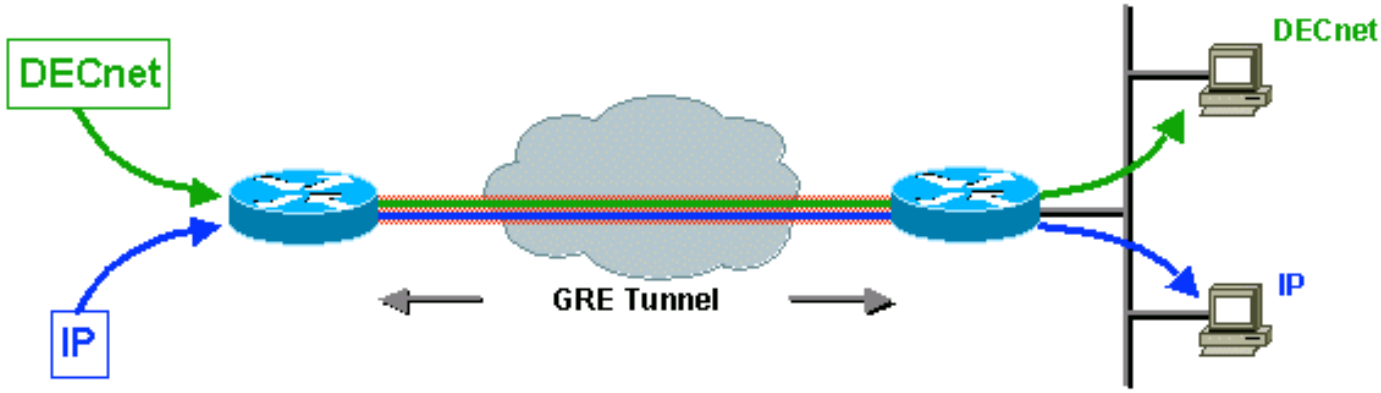
터널 패킷

IPv4	GRE	IPv4	TCP	Telnet

- IPv4는 전송 프로토콜입니다.
- GRE는 캡슐화 프로토콜입니다.
- IPv4는 승객 프로토콜입니다.

다음 예에서는 승객 프로토콜인 IPv4와 DECnet를 캐리어 프로토콜인 GRE로 캡슐화하는 과정을 보여줍니다.

다음은 이미지에 표시된 대로 캐리어 프로토콜이 여러 개의 패신저 프로토콜을 캡슐화할 가능성을 보여줍니다.



네트워크 관리자는 IPv4 백본으로 분리된 두 개의 비 IPv4 네트워크가 인접하지 않을 때 터널링을 고려합니다.

인접하지 않은 네트워크에서 DECnet가 실행되는 경우 관리자는 백본에 DECnet를 설정하여 두 네트워크를 연결하거나 연결하지 않을 수 있습니다.

또한 관리자는 IPv4 네트워크 성능에 방해가 될 수 있으므로 DECnet 라우팅에 백본 대역폭이 사용되는 것을 허용하지 않습니다.

이때 가능한 대안이 IPv4 백본에서 DECnet를 터널링하는 것입니다. 터널 솔루션은 IPv4 내부에서 DECnet 패킷을 캡슐화하고, 이 패킷을 백본을 통해 터널 엔드포인트로 전송합니다. 그러면 터널 엔드포인트에서 캡슐화가 제거되고 DECnet 패킷이 DECnet를 통해 대상으로 라우팅됩니다.

다른 프로토콜 내부에서 트래픽을 캡슐화하면 다음과 같은 이점이 있습니다.

- 엔드포인트가 전용 주소(RFC 1918)를 사용하고 백본은 이러한 주소에 대한 라우팅을 지원하지 않습니다.
- WAN 또는 인터넷에서 VPN(Virtual Private Network)이 허용됩니다.
- 인접하지 않은 멀티프로토콜 네트워크가 단일 프로토콜 백본에서 결합됩니다.
- 백본 또는 인터넷에서 트래픽이 암호화됩니다.

이후에는 IPv4가 패신저 프로토콜로, IPv4가 전송 프로토콜로 사용됩니다.

터널 인터페이스와 관련한 고려 사항

다음은 터널링 시 고려해야 할 사항입니다.

- Cisco IOS® Release 11.1에는 GRE 터널의 고속 스위칭이 도입되었고, 버전 12.0에는 CEF 스위칭이 도입되었습니다.
- 버전 12.2(8)T에는 멀티포인트 GRE 터널을 위한 CEF 스위칭이 도입되었습니다.
- 프로세스 스위칭만 지원되는 경우 이전 버전의 Cisco IOS®에서 터널 엔드포인트에서의 캡슐

화와 역캡슐화의 속도가 느렸습니다.

- 패킷을 터널링할 경우 보안 및 토폴로지 문제가 있습니다. 터널이 ACL(Access Control List) 및 방화벽을 우회할 수 있습니다.
- 방화벽을 통과하여 터널링하는 경우, 터널링되는 패신저 프로토콜을 우회합니다. 따라서 승객 프로토콜에 정책을 적용하려면 터널 엔드포인트에 방화벽 기능을 포함하는 것이 좋습니다.
- 터널링은 레이턴시가 증가하기 때문에 타이머가 제한된 전송 프로토콜(예: DECnet)에 문제를 일으킵니다.
- 여러 환경에서 서로 다른 속도 링크(예: 고속 FDDI 링과 느린 9600bps 전화선)를 사용한 터널링은 패킷 재정렬 문제를 유발합니다. 혼합된 미디어 네트워크에서는 일부 승객 프로토콜이 제대로 작동하지 않습니다.
- 포인트 투 포인트 터널은 물리적 링크의 대역폭을 사용할 수 있습니다. 여러 포인트 투 포인트 터널에서 라우팅 프로토콜을 실행하는 경우, 각 터널 인터페이스가 대역폭을 갖고 터널이 실행되는 물리적 인터페이스도 대역폭을 갖습니다. 예를 들어, 10Mb 링크에서 실행되는 100개의 터널이 있는 경우 터널 대역폭을 100Kb로 설정할 수 있습니다. 터널의 기본 대역폭은 9Kb입니다.
- 라우팅 프로토콜은 실제 링크 상의 터널을 선호합니다. 터널이 실제로는 홉이 더 많고 다른 경로보다 비용이 더 높은데도 언뜻 보기에는 가장 비용이 낮고 한 개의 홉 링크처럼 보이기 때문입니다. 이 문제는 라우팅 프로토콜을 올바르게 설정하여 완화됩니다. 서로 다른 라우팅 프로토콜은 물리적 인터페이스에서 실행하는 것보다 터널 인터페이스에서 실행하는 것이 좋습니다.
- 재귀 라우팅의 문제는 터널 대상에 대한 올바른 정적 경로를 설정하여 방지합니다. 재귀 경로란 터널 대상에 이르는 가장 적합한 경로가 터널 자체를 통과하는 경우를 말합니다. 이 경우 터널 인터페이스가 가동되었다가 중단됩니다. 재귀 라우팅 문제가 발생하면 이 오류가 표시됩니다.

```
%TUN-RECURDOWN Interface Tunnel 0
temporarily disabled due to recursive routing
```

터널 엔드포인트에서 PMTUD에 참여하는 라우터

라우터가 터널의 엔드포인트일 경우 서로 다른 두 가지 PMTUD 역할을 합니다.

- 첫 번째 역할에서는 라우터가 호스트 패킷의 전달자가 됩니다. PMTUD 처리를 위해 라우터는 원래 데이터 패킷의 DF 비트와 패킷 크기를 확인하고 필요한 경우 적절한 조치를 취해야 합니다.
- 두 번째 역할은 라우터가 터널 패킷 내부에서 원래 IPv4 패킷을 캡슐화한 후에 수행됩니다. 이 단계에서 라우터는 PMTUD와 관련하여 터널 IPv4 패킷에 대해 호스트처럼 작동합니다.

라우터가 첫 번째 역할(호스트 IPv4 패킷을 전달하는 라우터)을 하는 경우 이 역할은 라우터가 터널 패킷 내부에서 호스트 IPv4 패킷을 캡슐화하기 전에 수행됩니다.

라우터가 호스트 패킷의 전달자로 참여하는 경우 다음 작업을 수행하게 됩니다.

- DF 비트가 설정되어 있는지 확인.
- 터널에서 수용할 수 있는 패킷 크기 확인.
- 프래그먼트화(패킷이 너무 크고 DF 비트가 설정되지 않은 경우), 프래그먼트 캡슐화 및 전송
- 패킷 삭제(패킷이 너무 크고 DF 비트가 설정된 경우), 발신 장치에 ICMP 메시지 전송.
- 캡슐화(패킷이 너무 크지 않은 경우) 및 전송.

일반적으로 캡슐화를 수행한 다음 프래그먼트화를 선택(두 개의 캡슐화 프래그먼트 전송)하거나 프래그먼트화를 수행한 다음 캡슐화를 수행하도록 선택(두 개의 캡슐화된 프래그먼트 전송)할 수 있습니다.

PMTUD와 예시 네트워크를 통과하는 패킷의 상호 작용을 보여주는 두 가지 예시를 이 섹션에서 자세히 설명합니다.

첫 번째 예제에서는 터널 소스에서 라우터가 전달 라우터의 역할을 하는 경우 패킷에 어떤 일이 발생하는지를 보여줍니다.

PMTUD 처리를 위해 라우터가 원래 데이터 패킷의 DF 비트와 패킷 크기를 확인하고 적절한 조치를 취해야 합니다.

이 예제에서는 터널에 GRE 캡슐화를 사용합니다. GRE는 캡슐화 전에 프래그먼트화를 수행합니다.

이후 예제에는 캡슐화 후에 프래그먼트화가 수행되는 시나리오가 나와 있습니다.

예제 1의 경우 DF 비트가 설정되지 않았고(DF = 0) GRE 터널 IPv4 MTU가 1476(1500-24)입니다.

예 1

1. 터널 소스에 있는 전달 라우터가 전송 호스트로부터 DF 비트가 지워진(DF = 0) 1500바이트 데이터그램을 수신합니다.

이 데이터그램은 20바이트 IP 헤더와 1480바이트 TCP 페이로드로 구성되어 있습니다.

IPv4	1480바이트 TCP + 데이터

2. GRE 오버헤드(24바이트)가 추가된 후 패킷이 IPv4 MTU에 비해 너무 크기 때문에, 전달 라우터가 데이터그램을 1476바이트(20바이트 IPv4 헤더 + 1456바이트 IPv4 페이로드)와 44바이트(20바이트 IPv4 헤더 + 24바이트 IPv4 페이로드)의 두 프래그먼트로 분할합니다.

GRE 캡슐화가 추가된 후에 패킷 크기는 발신 물리적 인터페이스 MTU보다 크지 않게 됩니다.

IP_0	1456바이트 TCP + 데이터

IP_1	24바이트 데이터

3. 전달 라우터가 원래 IPv4 데이터그램의 각 프래그먼트에 GRE 캡슐화(4바이트 GRE 헤더와 20바이트 IPv4 헤더 포함)를 추가합니다.

이제 두 IPv4 데이터그램은 길이가 각각 1500바이트와 68바이트이고, 프래그먼트가 아닌 개별 IPv4 데이터그램으로 표시됩니다.

IPv4	GRE	IP_0	1456바이트 TCP + 데이터
IPv4	GRE	IP_1	24바이트 데이터

4. 터널 대상 라우터가 원래 데이터그램의 각 프래그먼트에서 GRE 캡슐화를 제거합니다. 따라서 길이가 1476바이트와 24바이트인 2개의 IPv4 프래그먼트가 남게 됩니다.

이러한 IPv4 데이터그램 프래그먼트는 이 라우터에 의해 수신 호스트에 개별적으로 전달됩니다.

IP_0	1456바이트 TCP + 데이터
IP_1	24바이트 데이터

5. 수신 호스트가 이 두 개의 프래그먼트를 원래 데이터그램으로 재조합합니다.

IPv4	1480바이트 TCP + 데이터

예시 2는 네트워크 토폴로지 상황에서 전달 라우터의 역할을 보여줍니다.

라우터가 전달 라우터라는 동일한 역할을 하지만, 이번에는 DF 비트가 설정되어 있습니다(DF = 1).

예 2

1. 터널 소스에 있는 전달 라우터가 전송 호스트로부터 DF가 1로 설정된 1500바이트 데이터그램을 수신합니다.

IPv4	1480바이트 TCP + 데이터

2. DF 비트가 설정되어 있고 데이터그램 크기(1500바이트)가 GRE 터널 IPv4 MTU(1476)보다 크므로, 라우터가 데이터그램을 삭제하고 데이터그램의 소스에 "ICMP fragmentation needed but DF bit set" 메시지를 전송합니다.

ICMP 메시지는 발신 장치에 MTU가 1476임을 알립니다.

IPv4	ICMP MTU 1476

3. 전송 호스트가 ICMP 메시지를 수신하고, 1476바이트의 IPv4 데이터그램을 사용하여 원래 데이터를 재전송합니다.

IPv4	1456바이트 TCP + 데이터

4. 이 IPv4 데이터그램 길이(1476바이트)가 이제 GRE 터널 IPv4 MTU의 값과 같기 때문에, 라우터는 IPv4 데이터 그램에 GRE 캡슐화를 추가합니다.

IPv4	GRE	IPv4	1456바이트 TCP + 데이터

5. 터널 대상에 있는 수신 라우터가 IPv4 데이터그램의 GRE 캡슐화를 제거하고 이를 수신 호스트에 전송합니다.

IPv4	1456바이트 TCP + 데이터

이는 라우터가 PMTUD와 관련하여 터널 IPv4 패킷에 두 번째 역할인 전송 호스트 역할을 할 때 발생합니다.

이 역할은 라우터가 터널 패킷 내부에서 원래 IPv4 패킷을 캡슐화한 후에 수행됩니다.

참고: 기본적으로 라우터는 자신이 생성하는 GRE 터널 패킷에 대해 PMTUD를 수행하지 않습니다. 이 **tunnel path-mtu-discovery** 명령을 사용하여 GRE-IPv4 터널 패킷에 대한 PMTUD를 켤 수 있습니다.

예제 3은 호스트가 GRE 터널 인터페이스에서 IPv4 MTU에 적합한 크기의 작은 IPv4 데이터그램을 전송할 때 어떤 일이 발생하는지 보여줍니다.

이 경우 DF 비트는 설정되거나 지워질 수 있습니다(1 또는 0).

GRE 터널 인터페이스에는 이 명령이 구성되어 있지 않으므로 **tunnel path-mtu-discovery** 라우터는 GRE-IPv4 패킷에서 PMTUD를 사용하지 않습니다.

예 3

1. 터널 소스에 있는 전달 라우터가 전송 호스트로부터 1476바이트 데이터그램을 수신합니다.

IPv4	1456바이트 TCP + 데이터

2. 이 라우터가 1500바이트 GRE IPv4 데이터그램을 받기 위해 GRE 내에서 1476바이트 IPv4 데이터그램을 캡슐화합니다.

GRE IPv4 헤더의 DF 비트가 지워집니다(DF = 0). 그런 다음 이 라우터는 이 패킷을 터널 대상으로 전달합니다.

IPv4	GRE	IPv4	1456바이트 TCP + 데이터

3. 터널 소스와 대상 사이에 링크 MTU가 1400인 라우터가 있다고 가정하겠습니다.

DF 비트가 지워졌기 때문에(DF = 0) 이 라우터는 터널 패킷을 프래그먼트화합니다.

이 예시에서는 가장 바깥쪽에 있는 IPv4가 프래그먼트화되므로 GRE와 내부 IPv4, TCP 헤더만 첫 번째 프래그먼트에 표시된다는 점을 기억하십시오.

IP_0	GRE	IP	1352바이트 TCP + 데이터
IP_1	104바이트 데이터		

4. 터널 대상 라우터가 GRE 터널 패킷을 재조합해야 합니다.

IP	GRE	IP	1456바이트 TCP + 데이터

5. GRE 터널 패킷이 재조합되면 라우터는 이제 GRE IPv4 헤더를 제거하고 원래 IPv4 데이터그램을 전송합니다.

IPv4	1456바이트 TCP + 데이터

예시 4에서는 라우터가 PMTUD와 관련하여 터널 IPv4 패킷에 대해 전송 호스트 역할을 하면 어떤 일이 발생하는지 보여줍니다.

이번에는 원래 IPv4 헤더에 DF 비트가 설정되고(DF = 1), 내부 IPv4 헤더에서 외부(GRE + IPv4) 헤더로 DF 비트가 복사되도록 명령이 **tunnel path-mtu-discovery** 구성되었습니다.

예 4

1. 터널 소스에 있는 전달 라우터가 전송 호스트로부터 DF가 1로 설정된 1476바이트 데이터그램을 수신합니다.

IPv4	1456바이트 TCP + 데이터

2. 이 라우터가 1500바이트 GRE IPv4 데이터그램을 받기 위해 GRE 내에서 1476바이트 IPv4 데이터그램을 캡슐화합니다.

원래 IPv4 데이터그램에 DF 비트가 설정되어 있으므로 이 GRE IPv4 헤더에 DF 비트가 설정됩니다(DF = 1).

그런 다음 이 라우터는 이 패킷을 터널 대상으로 전달합니다.

IPv4	GRE	IPv4	1456바이트 TCP

3. 다시, 터널 소스와 대상 사이에 링크 MTU가 1400인 라우터가 있다고 가정하겠습니다.

DF 비트가 설정되어 있기 때문에(DF = 1) 이 라우터는 터널 패킷을 프래그먼트화하지 않습니다.

이 라우터는 패킷을 삭제하고 패킷의 소스 IPv4 주소인 터널 소스 라우터에 ICMP 오류 메시지를 전송해야 합니다.

IPv4	ICMP MTU 1400

4. 터널 소스에 있는 전달 라우터가 이 "ICMP" 오류 메시지를 수신하고 GRE 터널 IPv4 MTU를 1376(1400-24)으로 낮춥니다.

다음번에 전송 호스트가 1476바이트 IPv4 패킷의 데이터를 재전송할 경우 이 패킷이 너무 클 수 있기 때문에, 이 라우터는 발신 장치에 MTU 값이 1376이라는 "ICMP" 오류 메시지를 보냅니다.

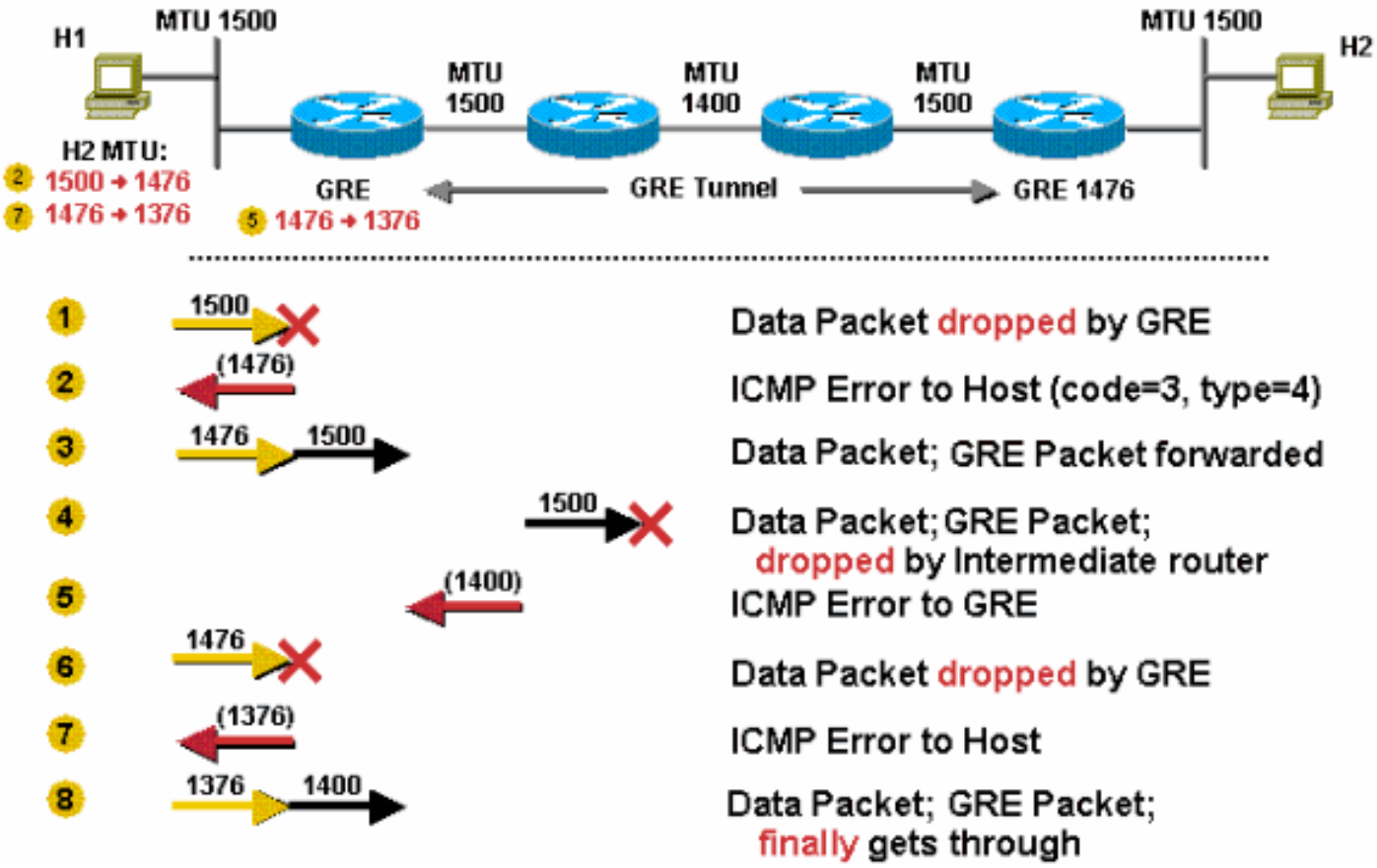
그러면 전송 호스트가 데이터를 재전송할 때 1376바이트 IPv4 패킷의 데이터를 보내고, 이 패킷은 GRE 터널을 통과하여 수신 호스트에 전달됩니다.

예 5

이 예시에서는 GRE 프래그먼트화를 설명합니다. GRE에 대해 캡슐화 전에 먼저 프래그먼트화를 수행한 다음 데이터 패킷에 대해 PMTUD를 수행합니다. 또한 IPv4 패킷이 GRE에 의해 캡슐화될

때 DF 비트가 복사되지 않습니다.

DF 비트는 설정되지 않습니다. GRE 터널 인터페이스 IPv4 MTU는 기본적으로 물리적 인터페이스 IPv4 MTU보다 24바이트 더 작기 때문에, 이미지에 표시된 대로 GRE 인터페이스 IPv4 MTU는 1476입니다.



1. 발신 장치가 1500바이트 패킷(20바이트 IPv4 헤더 + 1480바이트 TCP 페이로드)을 전송합니다.
2. GRE 터널의 MTU가 1476이므로, 1500바이트 패킷은 두 개의 IPv4 프래그먼트(1476바이트와 44바이트)로 나뉩니다. 각 프래그먼트에는 GRE 헤더의 24바이트가 추가될 것으로 예상됩니다.
3. GRE 헤더의 24바이트가 각 IPv4 프래그먼트에 추가됩니다. 이제 프래그먼트는 1500(1476 + 24)바이트 및 68(44 + 24)바이트입니다.
4. 두 IPv4 프래그먼트를 포함하는 GRE + IPv4 패킷이 GRE 터널 피어 라우터에 전달됩니다.
5. GRE 터널 피어 라우터가 두 패킷에서 GRE 헤더를 제거합니다.
6. 이 라우터가 두 패킷을 대상 호스트에 전달합니다.
7. 대상 호스트가 IPv4 프래그먼트를 원래 IPv4 데이터그램으로 다시 재조합합니다.

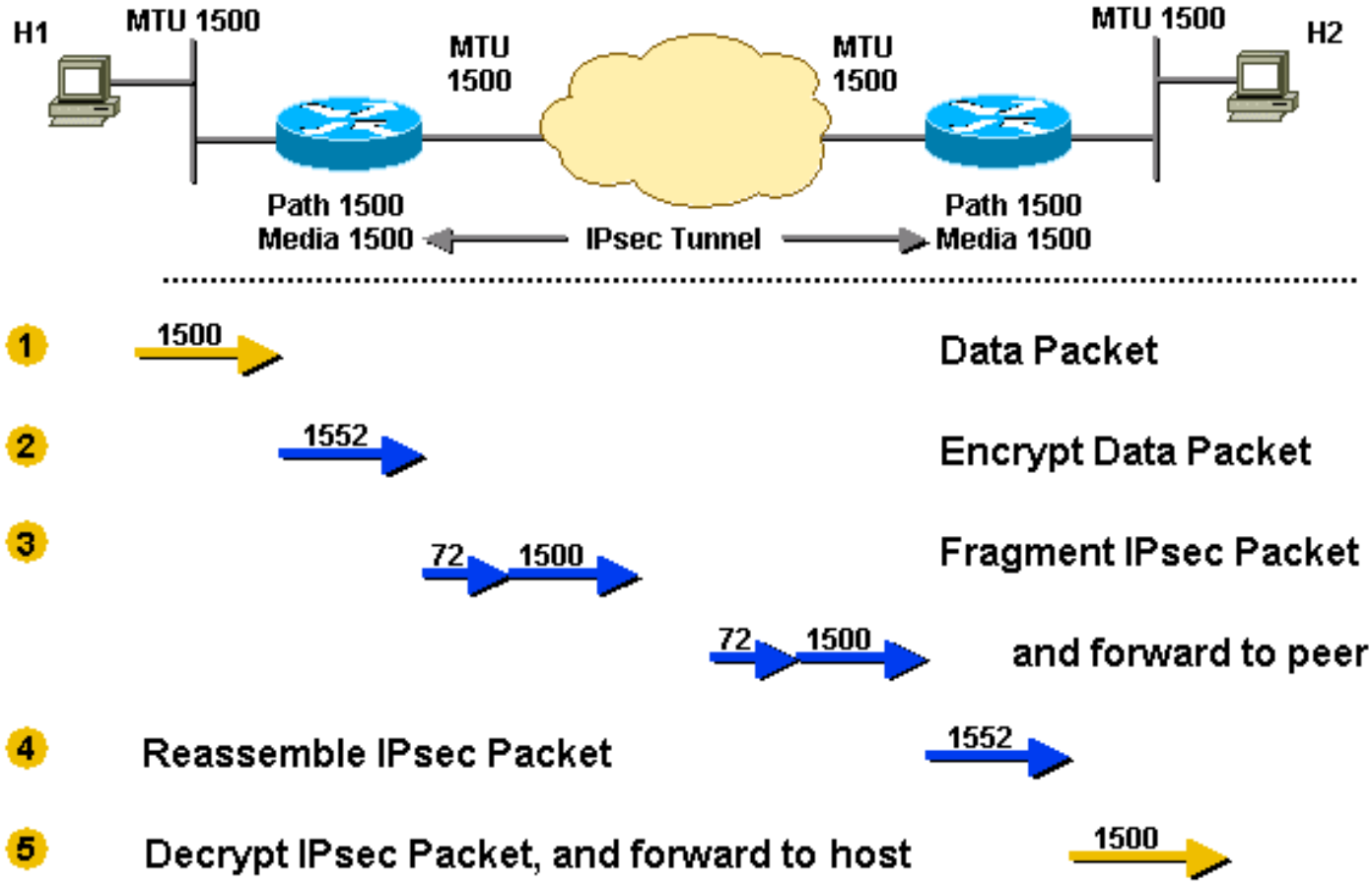
예 6

이 예시는 예시 5와 유사하지만, 이번에는 DF 비트가 설정되어 있습니다. 라우터는 명령을 사용하여 GRE + IPv4 터널 패킷에 PMTUD를 수행하도록 **tunnel path-mtu-discovery**구성되며, DF 비트가 원래 IPv4 헤더에서 GRE IPv4 헤더로 복사됩니다.

라우터가 GRE + IPv4 패킷에 대한 ICMP 오류를 수신하면 GRE 터널 인터페이스의 IPv4 MTU를 줄

입니다.

기본적으로 GRE 터널 IPv4 MTU가 물리적 인터페이스 MTU보다 24바이트 적게 설정되므로 여기서 GRE IPv4 MTU는 1476입니다. 이미지에 표시된 것처럼 GRE 터널 경로에는 1400 MTU 링크가 있습니다.



1. 라우터가 1500바이트 패킷(20바이트 IPv4 헤더 + 1480TCP 페이로드)을 수신한 다음 패킷을 삭제합니다. 라우터가 패킷을 삭제한 이유는 패킷이 GRE 터널 인터페이스의 IPv4 MTU(1476)보다 크기 때문입니다.
2. 라우터는 다음 홉 MTU가 1476임을 알려주는 ICMP 오류를 발신 장치에 보냅니다. 호스트는 라우팅 테이블에 이 정보를 대개 대상의 호스트 경로로 기록합니다.
3. 전송 호스트가 데이터를 재전송할 때 1476바이트 패킷 크기를 사용합니다. GRE 라우터가 GRE 캡슐화의 24바이트를 추가하고 1500바이트 패킷을 내보냅니다.
4. 1500바이트 패킷은 1400바이트 링크를 통과할 수 없으므로 중간 라우터에 의해 삭제됩니다.
5. 중간 라우터는 다음 홉 MTU가 1400이라는 ICMP(type = 3, code = 4)를 GRE 라우터에 보냅니다. GRE 라우터는 패킷을 1376(1400-24)으로 줄이고, GRE 인터페이스의 내부 IPv4 MTU 값을 설정합니다. 이 변경은 명령을 사용할 때만 볼 수 **debug tunnel** 있으며 명령의 출력에서는 볼 수 **show ip interface tunnel<#>** 없습니다.
6. 다음번에 호스트가 1476바이트 패킷을 다시 전송하면 GRE 라우터가 패킷을 삭제합니다. 패킷 크기가 GRE 터널 인터페이스의 현재 IPv4 MTU(1376)보다 크기 때문입니다.
7. GRE 라우터는 다음 홉 MTU가 1376이라는 또 다른 ICMP(type = 3, code = 4)를 발신 장치에 보내고, 호스트가 현재 정보를 새 값으로 업데이트합니다.
8. 호스트는 데이터를 다시 전송합니다. 그러나 이제 크기가 더 작아진 1376바이트 패킷에 데이터가 있기 때문에 GRE는 24바이트 캡슐화를 추가한 다음 패킷을 전달합니다. 이번에는 패킷

이 GRE 터널 피어에 도달하고, 여기서 패킷이 역캡슐화된 다음 대상 호스트에 전송됩니다.

참고: **tunnel path-mtu-discovery** 이 시나리오에서 전달 라우터에 명령이 구성되지 않았고 GRE 터널을 통해 전달된 패킷에 DF 비트가 설정된 경우, 호스트 1은 여전히 TCP/IPv4 패킷을 호스트 2로 전송하는 데 성공하지만, 1400 MTU 링크에서 중간에 프래그먼트화됩니다. 또한 GRE 터널 피어가 패킷을 역캡슐화하고 전달하려면 먼저 패킷을 재조합해야 합니다.

순수 IPsec 터널 모드

IPv4sec(IPv4 Security) 프로토콜은 IPv4 네트워크에서 전송되는 정보에 프라이버시, 무결성 및 진본성을 부여하는 표준 기반 방법입니다.

IPv4sec는 IPv4 네트워크 레이어 암호화를 제공합니다. 그리고 하나 이상의 IPv4 헤더(터널 모드)를 추가하기 때문에 IPv4 패킷의 길이를 늘립니다.

추가되는 헤더의 길이는 IPv4sec 설정 모드에 따라 다르지만 패킷당 최대 58바이트 (ESP(Encapsulating Security Payload) 및 ESPauth(ESP authentication))를 초과하지 않습니다.

IPv4sec에는 두 가지 모드 즉, 터널 모드와 전송 모드가 있습니다.

1. 터널 모드가 기본 모드입니다. 터널 모드를 사용하면 원래 IPv4 패킷 전체가 보호(암호화됨 또는 인증됨, 또는 둘 다)되고 IPv4sec 헤더 및 트레일러에 의해 캡슐화됩니다. 그런 다음 IPv4sec 엔드포인트(피어)를 소스와 대상으로 지정하는 새 IPv4 헤더가 패킷에 추가됩니다. 터널 모드는 모든 유니캐스트 IPv4 트래픽에 사용할 수 있으며, IPv4sec가 IPv4sec 피어 뒤에 있는 호스트의 트래픽을 보호하는 경우에는 터널 모드를 사용해야 합니다. 예를 들어 보호되는 한 네트워크의 호스트가 보호되는 다른 네트워크의 호스트에 IPv4sec 피어를 통해 패킷을 전송하는 VPN(Virtual Private Network)에는 터널 모드가 사용됩니다. VPN을 통해 IPv4sec "터널"은 IPv4sec 피어 라우터 간의 IPv4 트래픽을 암호화하여 호스트 간의 IPv4 트래픽을 보호합니다.
2. 전송 모드(변환 정의에서 하위 명령**mode transport**으로 구성)를 사용하면 원래 IPv4 패킷의 페이로드만 보호(암호화, 인증 또는 둘 다)됩니다. 페이로드는 IPv4sec 헤더와 트레일러로 캡슐화됩니다. 원래 IPv4 헤더는 IPv4 프로토콜 필드가 ESP (50)으로 변경된다는 점을 제외하고 그대로 유지됩니다. 원래 프로토콜 값은 패킷이 암호 해독될 때 값이 복원되도록 IPv4sec 트레일러에 저장됩니다. 전송 모드는 보호되는 IPv4 트래픽이 IPv4sec 피어 자체 사이에 있는 경우와 패킷의 소스 IPv4 주소와 대상 IPv4 주소가 IPv4sec 피어 주소와 동일한 경우에만 사용됩니다. 일반적으로 IPv4sec 전송 모드는 IPv4 데이터 패킷을 처음 캡슐화할 때 다른 터널링 프로토콜(예: GRE)이 사용된 다음 GRE 터널 패킷을 보호할 때 IPv4sec가 사용되는 경우에만 사용됩니다.

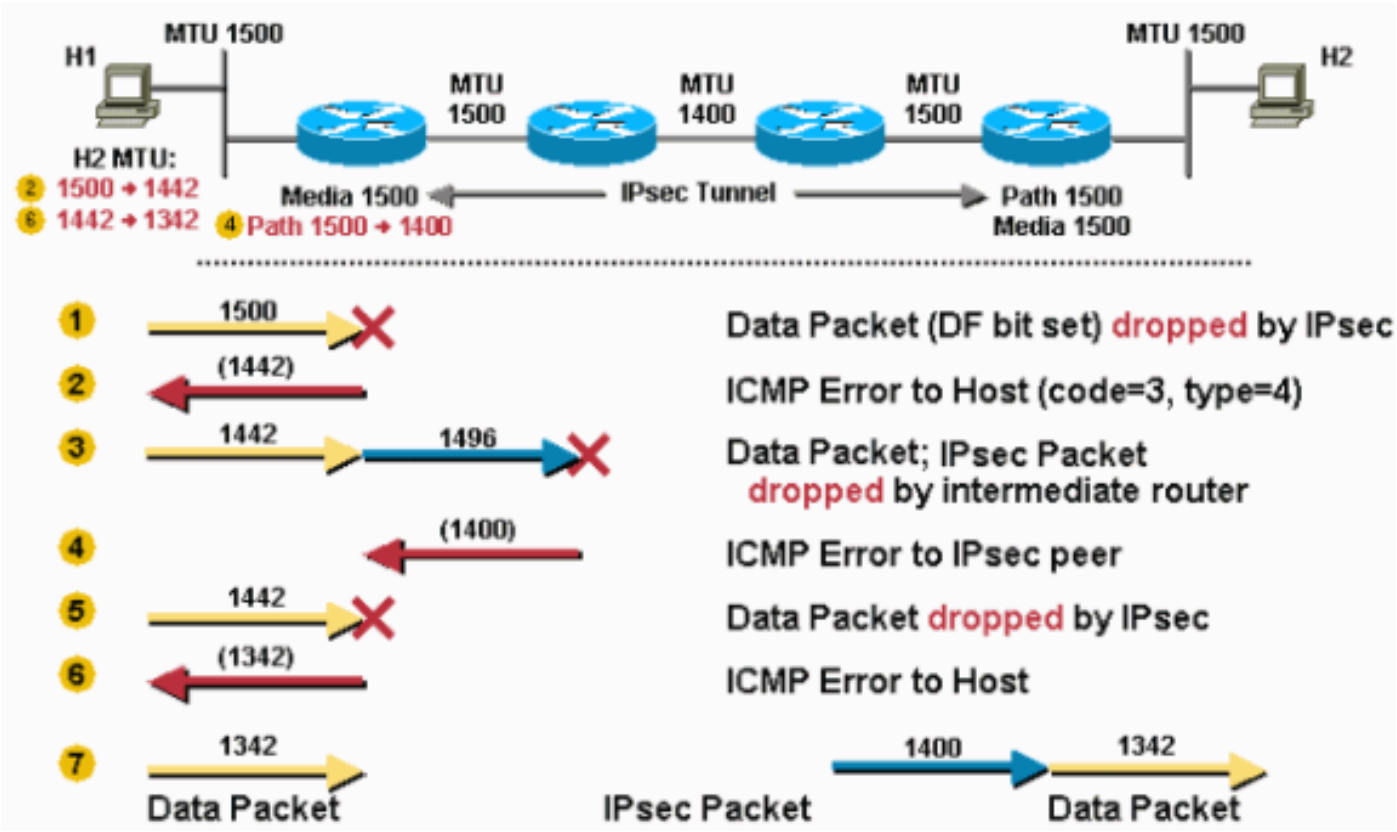
IPv4sec는 항상 데이터 패킷과 자체 패킷에 대해 PMTUD를 수행합니다. IPv4sec IPv4 패킷에 대한 PMTUD 처리를 수정할 수 있는 IPv4sec 컨피그레이션 명령이 있습니다. IPv4sec는 데이터 패킷 IPv4 헤더에서 DF 비트를 지우거나 설정하거나 IPv4sec IPv4 헤더에 복사할 수 있습니다. 이 기능을 "DF 비트 재정의 기능"이라고 합니다.

참고: IPv4sec를 사용하여 하드웨어 암호화를 수행할 경우 캡슐화한 후에는 프래그먼트화를 피하십시오. 하드웨어 암호화로 하드웨어에 따라 약 50Mb의 처리량이 발생하지만 IPv4sec

패킷이 프래그먼트화되는 경우 처리량의 50~90%가 손실됩니다. 이 손실이 발생하는 이유는 프래그먼트화된 IPv4sec 패킷이 재조합을 위해 프로세스 스위칭되고 암호 해독을 위해 하드웨어 암호화 엔진에 전달되기 때문입니다. 이 처리량 손실로 하드웨어 암호화 처리량이 소프트웨어 암호화의 성능 수준(2~10Mb)까지 떨어질 수 있습니다.

예 7

이 시나리오에서는 IPv4sec 프래그먼트화가 수행되는 방식을 보여줍니다. 이 시나리오에서 MTU는 전체 경로에서 1500입니다. 이 시나리오에서는 DF 비트가 설정되지 않습니다.



1. 라우터가 호스트 2를 대상으로 한 1500바이트 패킷(20바이트 IPv4 헤더 + 1480바이트 TCP 페이로드)을 수신합니다.
2. 1500바이트 패킷은 IPv4sec에 의해 암호화되고 52바이트의 오버헤드가 추가됩니다(IPv4sec 헤더, 트레일러 및 추가된 IPv4 헤더). 이제 IPv4sec는 1552바이트 패킷을 전송해야 합니다. 아웃바운드 MTU가 1500이므로 이 패킷은 프래그먼트화되어야 합니다.
3. IPv4sec 패킷에서 두 개의 프래그먼트가 생성됩니다. 프래그먼트화 중에 두 번째 프래그먼트에 20바이트 IPv4 헤더가 추가되고, 그 결과 1500바이트 프래그먼트와 72바이트 IPv4 프래그먼트가 생성됩니다.
4. IPv4sec 터널 피어 라우터가 프래그먼트를 수신하고, 추가 IPv4 헤더를 제거한 다음, IPv4 프래그먼트를 합쳐 다시 원래 IPv4sec 패킷으로 만듭니다. 그런 다음 IPv4sec가 이 패킷의 암호를 해독합니다.
5. 그러면 라우터가 원래 1500바이트 데이터 패킷을 호스트 2에 전달합니다.

예 8

이 예시는 예시 6과 유사합니다. 단, 이 경우에는 DF 비트가 원래 데이터 패킷에 설정되어 있고, IPv4sec 터널 피어 간의 경로에 다른 링크보다 MTU가 낮은 링크가 있습니다.

이 예시에서는 터널 엔드포인트에서 PMTUD에 참여하는 라우터 섹션에서 설명한 것처럼 IPv4sec 피어 라우터가 두 PMTUD 역할을 어떻게 수행하는지 설명합니다.

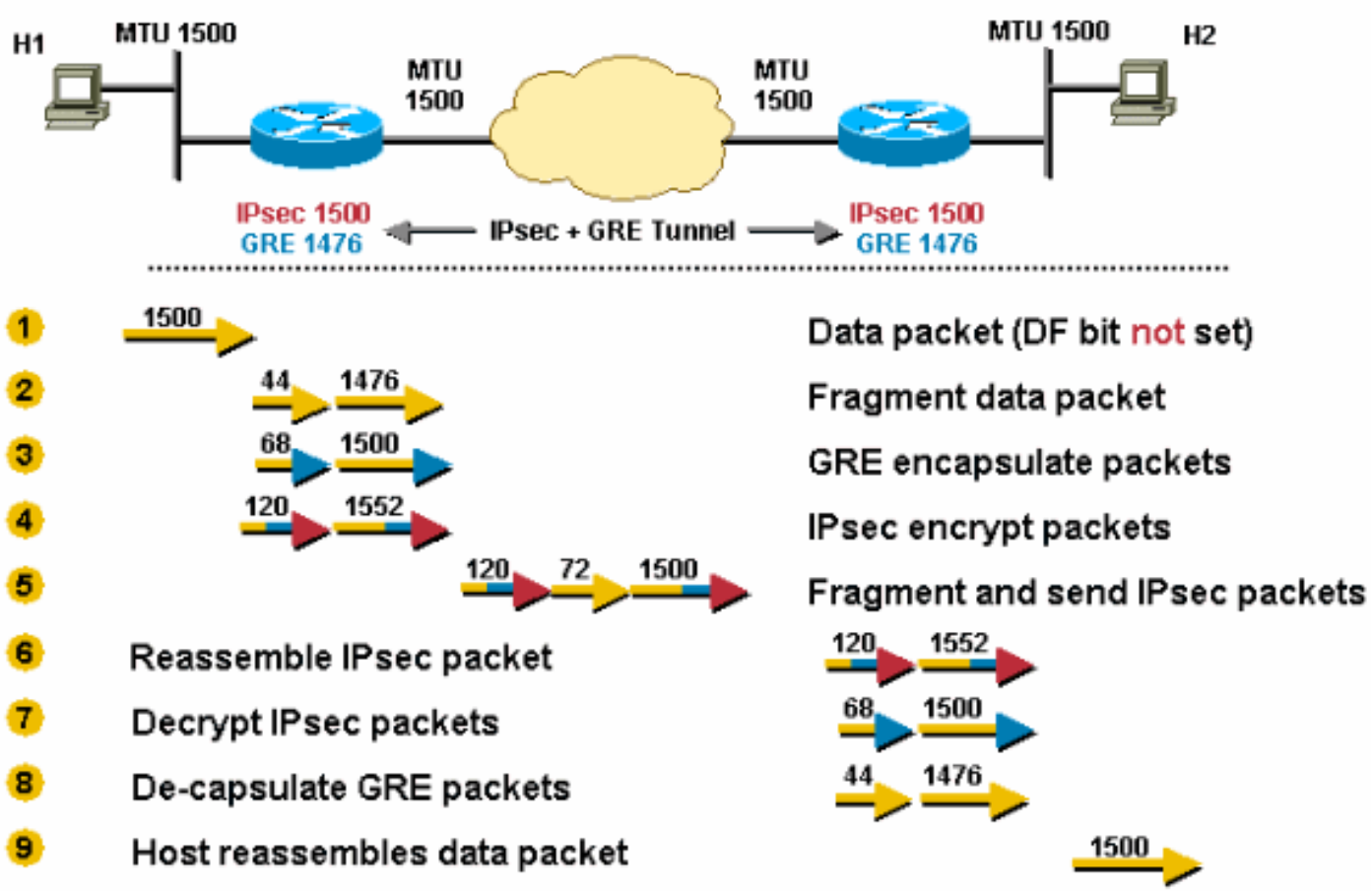
프래그먼트화에 필요하기 때문에 IPv4sec PMTU가 더 낮은 값으로 변합니다.

IPv4sec가 패킷을 암호화할 경우 내부 IPv4 헤더의 DF 비트가 외부 IPv4 헤더에 복사됩니다.

미디어 MTU 값과 PMTU 값은 IPv4sec SA(Security Association)에 저장됩니다.

미디어 MTU는 아웃바운드 라우터 인터페이스의 MTU를 기반으로 하고, PMTU는 IPv4sec 피어 간의 경로에 표시되는 최소 MTU를 기반으로 합니다.

이미지에 표시된 대로 IPv4sec는 패킷을 프래그먼트화하기 전에 캡슐화/암호화합니다.



1. 라우터가 1500바이트 패킷을 수신한 다음, IPv4sec 오버헤드가 추가되면 패킷이 PMTU(1500)보다 커지기 때문에 패킷을 삭제합니다.
2. 라우터는 다음 홉 MTU가 1442(1500 - 58 = 1442)임을 알리는 ICMP 메시지를 호스트 1에 전송합니다. 이 58바이트는 IPv4sec ESP 및 ESPauth를 사용할 때의 최대 IPv4sec 오버헤드입니다. 실제 IPv4sec 오버헤드는 이 값보다 7바이트 더 작을 수 있습니다. 호스트 1은 라우팅 테이블에 이 정보를 대개 대상(호스트 2)의 호스트 경로로 기록합니다.
3. 호스트 1은 호스트 2에 대한 PMTU를 1442로 줄입니다. 따라서 호스트 1은 데이터를 호스트 2로 다시 전송할 때 더 작은 패킷(1442바이트)을 전송하게 됩니다. 라우터가 1442바이트 패킷을 수신하고 IPv4sec가 52바이트의 암호화 오버헤드를 추가하므로, 그 결과로 생성되는

IPv4sec 패킷은 1496바이트입니다. 이 패킷은 헤더에 DF 비트가 설정되어 있으므로 1400바이트 MTU 링크를 사용하는 중간 라우터에 의해 삭제됩니다.
4. 패킷을 삭제한 중간 라우터가 IPv4sec 패킷의 발신 장치(첫 번째 라우터)에 다음 홉 MTU가 1400바이트임을 알리는 ICMP 메시지를 전송합니다. 이 값은 IPv4sec SA PMTU에 기록됩니다.
5. 다음번에 호스트 1이 1442바이트 패킷을 다시 전송할 경우(승인을 받지 못함) IPv4sec가 패킷을 삭제합니다. IPv4sec 오버헤드가 패킷에 추가되면 패킷이 PMTU(1400)보다 커지기 때문에 라우터가 패킷을 삭제합니다.
6. 라우터는 다음 홉 MTU가 이제 1342(1400 - 58 = 1342)임을 알리는 ICMP 메시지를 호스트 1에 전송합니다. 호스트 1이 이 정보를 다시 기록합니다.
7. 호스트 1이 다시 데이터를 전송할 때는 더 작은 크기의 패킷(1342)을 사용합니다. 이 패킷은 프래그먼트화가 필요하지 않으며 IPv4sec 터널을 거쳐 호스트 2에 도달합니다.

GRE와 IPv4sec 함께 사용

GRE 터널을 암호화하는 데 IPv4sec가 사용되는 경우 프래그먼트화와 PMTUD가 더 복잡하게 상호작용합니다.

IPv4sec와 GRE가 이렇게 함께 사용되는 이유는 IPv4sec가 IPv4 멀티캐스트 패킷을 지원하지 않아 IPv4sec VPN 네트워크에서 동적 라우팅 프로토콜을 실행할 수 없기 때문입니다.

GRE 터널이 멀티캐스트를 지원하기 때문에, IPv4sec에 의해 암호화될 수 있는 GRE IPv4 유니캐스트 패킷에서 동적 라우팅 프로토콜 멀티캐스트 패킷을 처음 캡슐화할 때 GRE 터널을 사용할 수 있습니다.

이 작업을 수행할 때 IPv4sec는 전송 모드로 GRE 위에 구축되는 경우가 많습니다. IPv4sec 피어와 GRE 터널 엔드포인트(라우터)가 동일하고 전송 모드가 20바이트의 IPv4sec 오버헤드를 줄여주기 때문입니다.

한 가지 흥미로운 사례는 IPv4 패킷이 두 개의 프래그먼트로 분할되고 GRE에 의해 캡슐화된 경우입니다.

이 경우 IPv4sec는 두 개의 독립적인 GRE + IPv4 패킷을 발견합니다. 일반적으로 기본 설정에서는 이러한 패킷 중 하나의 크기가 커서 암호화된 후 프래그먼트화되어야 합니다.

IPv4sec 피어가 암호 해독 전에 이 패킷을 재조합해야 합니다. 이렇게 전송 라우터에서 "이중 프래그먼트화"(GRE 전에 한 번, IPv4sec 후에 한 번)가 발생하여 레이턴시가 증가하고 처리량이 줄어듭니다.

재조합은 프로세스 스위칭이 수반되기 때문에 재조합이 발생할 때마다 수신 라우터의 CPU가 과부하 상태가 됩니다.

이 상황은 GRE와 IPv4sec의 오버헤드를 모두 고려할 정도로 GRE 터널 인터페이스의 "ip mtu"를 낮게 설정하면 방지할 수 있습니다(기본적으로 GRE 터널 인터페이스 "ip mtu"는 보내는 실제 인터페이스 MTU - GRE 오버헤드 바이트로 설정됨).

아래 테이블에는 보내는 물리적 인터페이스의 MTU가 1500이라고 가정할 때 각 터널/모드 조합에 권장되는 MTU 값이 나열되어 있습니다.

터널 조합	필요한 특정 MTU	권장되는 MTU
GRE + IPv4sec(전송 모드)	1440바이트	1400바이트
GRE + IPv4sec(터널 모드)	1420바이트	1400바이트

참고: 가장 일반적인 GRE + IPv4sec 모드 조합을 포함하기 때문에 MTU 값 1400이 권장됩니다. 또한 20바이트 또는 40바이트의 추가 오버헤드를 감안해야 한다는 단점도 크게 없습니다. 한 가지 값을 설정하고 기억하는 것이 더 쉽고, 이 값은 거의 모든 시나리오에 적용 가능합니다.

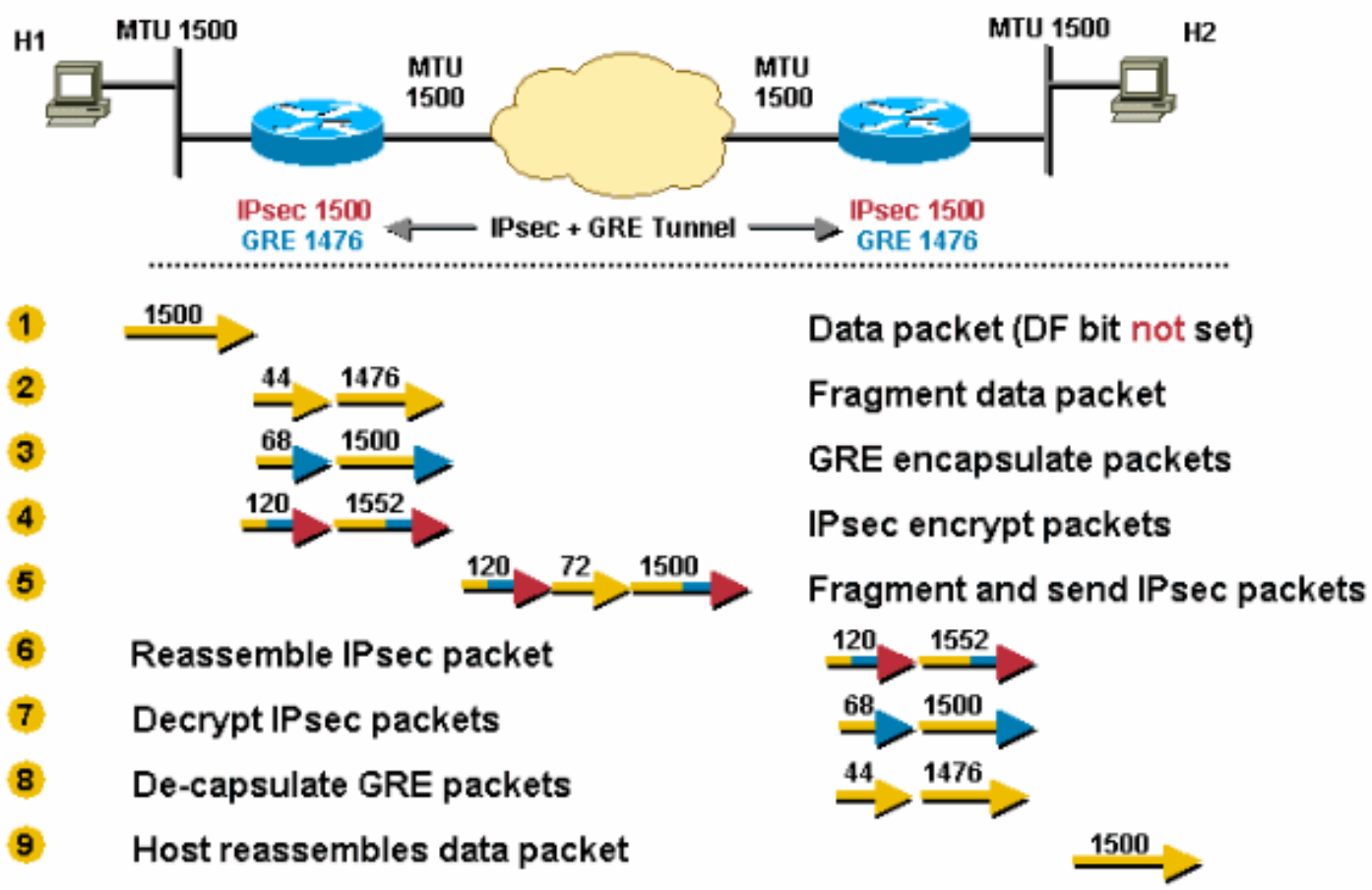
예 9

IPv4sec는 GRE 위에 구축됩니다. 보내는 물리적 MTU는 1500, IPv4sec PMTU는 1500, GRE IPv4 MTU는 1476(1500 - 24 = 1476)입니다.

따라서 TCP/IPv4 패킷은 GRE 전에 한 번, IPv4sec 후에 한 번, 총 두 번 프래그먼트화됩니다.

패킷은 GRE 캡슐화 전에 프래그먼트화되고 이러한 GRE 패킷 중 하나는 IPv4sec 암호화 후에 다시 프래그먼트화됩니다.

GRE 터널에 "ip mtu 1440"(IPv4sec 전송 모드) 또는 "ip mtu 1420"(IPv4sec 터널 모드)을 구성하면 이 시나리오에서 이중 프래그먼트화가 발생할 가능성이 사라집니다.



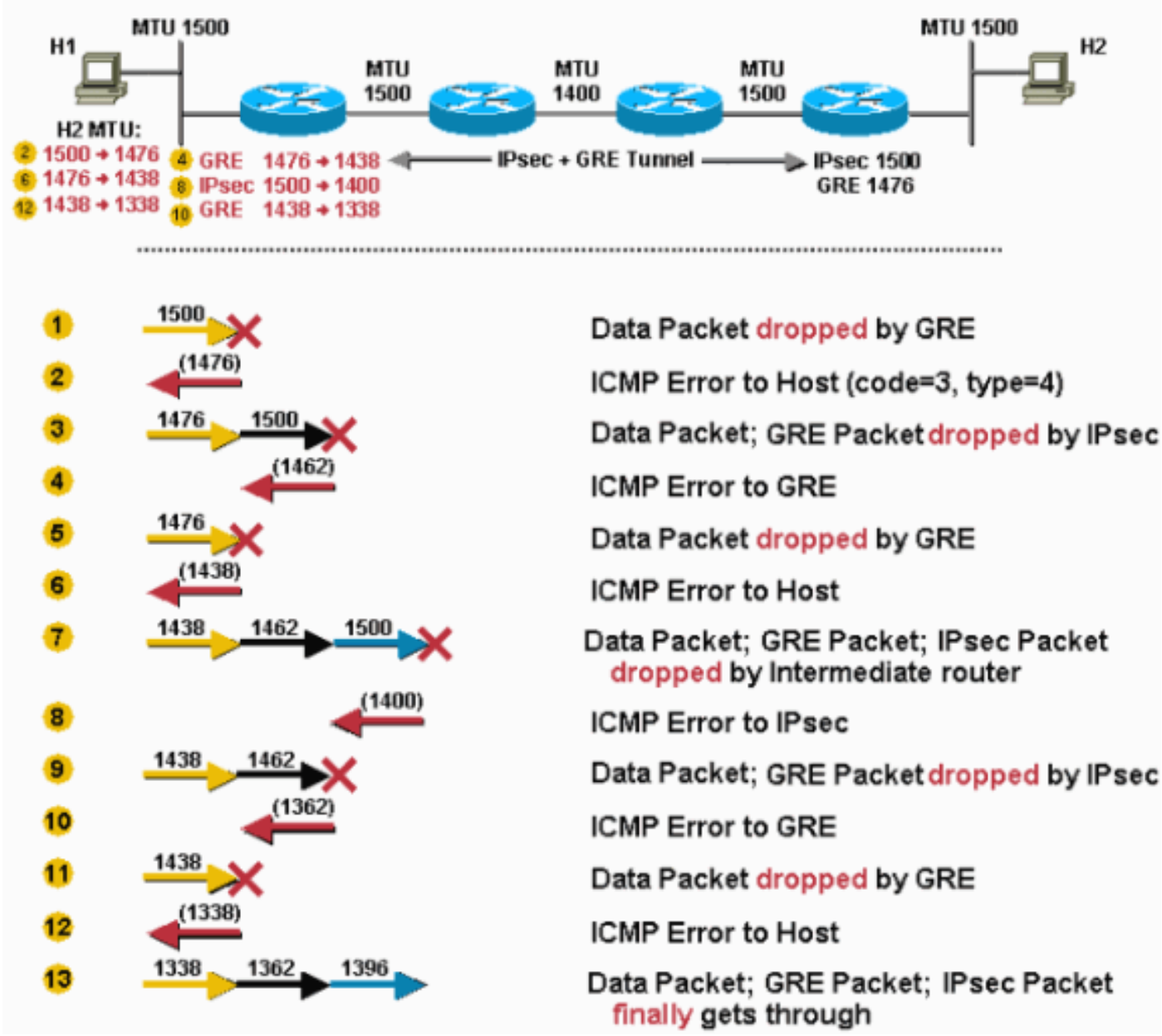
1. 라우터가 1500바이트 데이터그램을 수신합니다.
2. 캡슐화 전에 GRE가 1500바이트 패킷을 2개로 즉, 1476바이트(1500 - 24 = 1476)와 44바이트(24 데이터 + 20 IPv4 헤더)로 프래그먼트화합니다.

3. GRE가 IPv4 프래그먼트를 캡슐화하여 각 패킷에 24바이트가 추가됩니다. 그 결과 각각 1500바이트(1476 + 24 = 1500)와 68바이트(44 + 24)인 두 개의 GRE + IPv4sec 패킷이 생성됩니다.
4. IPv4sec가 두 개의 패킷을 암호화하고, 1552바이트 패킷과 120바이트 패킷을 제공하기 위해 두 패킷 각각에 캡슐화 오버헤드 52바이트(IPv4sec 터널 모드)를 더합니다.
5. 1552바이트 IPv4sec 패킷은 아웃바운드 MTU(1500)보다 크므로 라우터에 의해 프래그먼트화됩니다. 1552바이트 패킷은 1500바이트 패킷과 72바이트 패킷(52바이트 "페이로드" + 두 번째 프래그먼트의 추가 20바이트 IPv4 헤더)으로 분할됩니다. 3개의 패킷 즉, 1500바이트, 72바이트, 120바이트 패킷이 IPv4sec + GRE 피어에 전달됩니다.
6. 수신 라우터가 원래 1552바이트 IPv4sec + GRE 패킷을 받기 위해 두 개의 IPv4sec 프래그먼트(1500바이트 및 72바이트)를 재조합합니다. 120바이트 IPv4sec + GRE 패킷에 대해서는 아무것도 수행할 필요가 없습니다.
7. IPv4sec가 1500바이트 패킷과 68바이트 GRE 패킷을 받기 위해 1552바이트와 120바이트 IPv4sec + GRE 패킷의 암호를 모두 해독합니다.
8. GRE가 1476바이트 및 44바이트 IPv4 패킷 프래그먼트를 받기 위해 1500바이트 및 68바이트 GRE 패킷을 역캡슐화합니다. 이러한 IPv4 패킷 프래그먼트가 대상 호스트에 전달됩니다.
9. 호스트 2가 원래 1500바이트 IPv4 데이터그램을 받기 위해 이러한 IPv4 프래그먼트를 재조합합니다.

시나리오 10은 터널 경로에 MTU가 더 낮은 링크가 있다는 점을 제외하고 시나리오 8과 유사합니다. 이 시나리오는 호스트 1에서 호스트 2로 처음 전송되는 패킷에 가장 나쁜 시나리오입니다. 이 시나리오의 마지막 단계 이후에, 호스트 1이 호스트 2에 대한 올바른 PMTU를 설정하기 때문에 호스트 1과 호스트 2 간의 TCP 연결이 순조롭게 이루어집니다. 호스트 1과 다른 호스트 간의 TCP 흐름(IPv4sec + GRE 터널을 통해 연결 가능함)은 시나리오 10의 마지막 세 단계만 거치면 됩니다.

이 시나리오에서 **tunnel path-mtu-discovery** 명령은 GRE 터널에 구성되고 DF 비트는 호스트 1에서 시작되는 TCP/IPv4 패킷에 설정됩니다.

예 10



- 라우터가 1500바이트 패킷을 수신합니다. 이 패킷은 GRE에 의해 삭제됩니다. DF 비트가 설정되어 있고 GRE 오버헤드(24바이트)가 추가되면 패킷 크기가 아웃바운드 인터페이스 "ip mtu"를 초과하여 GRE가 패킷을 프래그먼트화하거나 전달할 수 없기 때문입니다.
- 라우터는 다음 홉 MTU가 1476(1500 - 24 = 1476)임을 알리기 위해 ICMP 메시지를 호스트 1에 전송합니다.
- 호스트 1이 호스트 2에 대한 PMTU를 1476으로 변경하고 패킷을 재전송할 때 더 작은 크기의 패킷을 전송합니다. GRE가 패킷을 캡슐화하고 IPv4sec에 1500바이트 패킷을 전달합니다. IPv4sec가 패킷을 삭제합니다. GRE가 내부 IPv4 헤더의 DF 비트(설정됨)를 복사했고 IPv4sec 오버헤드(최대 38바이트)가 추가되면서 패킷이 너무 커져 물리적 인터페이스를 벗어나 전달될 수 없기 때문입니다.
- IPv4sec는 다음 홉 MTU가 1462바이트(암호화 및 IPv4 오버헤드를 위해 최대 38바이트가 추가되기 때문)임을 나타내는 ICMP 메시지를 GRE에 전송합니다. GRE가 터널 인터페이스에 값 1438(1462 - 24)을 "ip mtu"로 기록합니다.

- 참고: 이 값 변경은 내부에 저장되며 **show ip interface tunnel<#>**명령 출력에서 볼 수 없습니다.

이 변경 사항은 명령을 사용하는 경우에만 **debug tunnel** 표시됩니다.

- 다음번에 호스트 1이 1476바이트 패킷을 재전송하면 GRE가 이 패킷을 삭제합니다.
- 라우터는 다음 홉 MTU가 1438임을 알리는 ICMP 메시지를 호스트 1에 보냅니다.
- 호스트 1이 호스트 2에 대한 PMTU를 낮추고 1438바이트 패킷을 재전송합니다. 이번에는 GRE가 패킷을 수락하고 캡슐화한 다음 암호화를 위해 IPv4sec에 전달합니다.
- IPv4sec 패킷이 중간 라우터에 전달되고, 중간 라우터의 아웃바운드 인터페이스 MTU가 1400이기 때문에 삭제됩니다.
- 중간 라우터는 다음 홉 MTU가 1400임을 나타내는 ICMP 메시지를 IPv4sec에 전송합니다. 이 값은 연결된 IPv4sec SA의 PMTU 값에 IPv4sec에 의해 기록됩니다.
- 호스트 1이 1438바이트 패킷을 재전송하면 GRE가 이 패킷을 캡슐화하고 IPv4sec에 전달합니다. IPv4sec가 자체 PMTU를 1400으로 변경했기 때문에 패킷을 삭제합니다.
- IPv4sec는 다음 홉 MTU가 1362임을 나타내는 ICMP 오류를 GRE에 전송하고, GRE는 값 1338을 내부에 기록합니다.
- 호스트 1이 원래 패킷을 재전송(승인을 받지 못했기 때문)하면 GRE가 이 패킷을 삭제합니다.
- 라우터는 다음 홉 MTU가 1338(1362 - 24바이트)임을 나타내는 ICMP 메시지를 호스트 1에 보냅니다. 호스트 1이 호스트 2에 대한 PMTU를 1338까지 줄입니다.
- 호스트 1이 1338바이트 패킷을 재전송하고, 결국 패킷이 호스트 2까지 안전하게 전송됩니다.

추가 권장 사항

터널 인터페이스**tunnel path-mtu-discovery** 에서 명령을 구성하면 GRE와 IPv4sec가 동일한 라우터에 구성된 경우 상호 작용하는데 도움이 될 수 있습니다.

명령이 **tunnel path-mtu-discovery** 구성되지 않으면 GRE IPv4 헤더에서 DF 비트가 항상 지워집니다.

그러면 캡슐화된 데이터 IPv4 헤더에 DF 비트가 설정되어 있더라도 GRE IPv4 패킷이 프래그먼트화될 수 있습니다. 캡슐화된 데이터 IPv4 헤더에 DF 비트가 설정되어 있으면 대개 패킷 프래그먼트화가 발생하지 않습니다.

GRE **tunnel path-mtu-discovery** 터널 인터페이스에 명령이 구성된 경우:

1. GRE가 데이터 IPv4 헤더의 DF 비트를 GRE IPv4 헤더에 복사합니다.
2. GRE IPv4 헤더에 DF 비트가 설정되고 앞으로 패킷이 IPv4sec 암호화를 거치면 물리적 발신 인터페이스의 IPv4 MTU에 비해 "너무 큰" 상태가 되는 경우, IPv4sec는 패킷을 삭제하고 GRE 터널에 IPv4 MTU 크기를 줄이라고 알립니다.
3. IPv4sec가 자체 패킷에 대해 PMTUD를 수행하고, IPv4sec PMTU가 변경(축소)될 경우 IPv4sec는 즉시 GRE에 알리지 않습니다. 그러나 큰 패킷이 또 나타나면 2단계의 프로세스가 발생합니다.
4. 이제 GRE의 IPv4 MTU가 더 작아졌습니다. 이 때문에 GRE는 DF 비트가 설정되고 너무 커진 데이터 IPv4 패킷을 모두 삭제하고 전송 호스트에 ICMP 메시지를 보냅니다.

이 **tunnel path-mtu-discovery** 명령은 GRE 인터페이스에서 해당 IPv4 MTU를 정적으로 설정하는 대신 동적으로 설정하는 데 **ip mtu** 유용합니다. 사실 두 가지 명령을 모두 사용하는 것이 좋습니다.

이 **ip mtu** 명령은 로컬 물리적 발신 인터페이스 IPv4 MTU를 기준으로 GRE 및 IPv4sec 오버헤드를 위한 공간을 제공하는 데 사용됩니다.

이 **tunnel path-mtu-discovery** 명령을 사용하면 IPv4sec 피어 간의 경로에 더 낮은 IPv4 MTU 링크가 있는 경우 GRE 터널 IPv4 MTU를 더 줄일 수 있습니다.

다음은 GRE + IPv4sec 터널이 구성된 네트워크에서 PMTUD 문제가 발생한 경우 수행할 수 있는 몇 가지 작업입니다.

가장 바람직한 방법부터 나열되었습니다.

1. PMTUD가 수행되지 않는 문제를 해결하십시오. 이러한 문제는 대부분 ICMP를 차단하는 라우터 또는 방화벽에 의해 발생합니다.
2. 라우터가 **ip tcp adjust-mss** TCP SYN 패킷의 TCP MSS 값을 줄이도록 터널 인터페이스에서 이 명령을 사용합니다. 이렇게 하면 PMTUD가 필요하지 않을 정도로 작은 패킷을 두 종단 호스트(TCP 발신 장치 및 수신 장치)에서 사용할 수 있습니다.
3. 라우터의 인그레스 인터페이스에 정책 라우팅을 사용하고, 데이터 IPv4 헤더가 GRE 터널 인터페이스에 도착하기 전에 데이터 IPv4 헤더의 DF 비트를 지우도록 경로 맵을 구성하십시오. 이렇게 하면 데이터 IPv4 패킷이 GRE 캡슐화 전에 프래그먼트화될 수 있습니다.
4. GRE 터널 인터페이스의 "ip mtu"를 아웃바운드 인터페이스 MTU와 동일하도록 늘리십시오. 이렇게 하면 데이터 IPv4 패킷이 먼저 프래그먼트화되기 전에 GRE 캡슐화될 수 있습니다. 그러면 GRE 패킷은 IPv4sec 암호화를 거친 후, 물리적 아웃바운드 인터페이스를 벗어나기 위해 프래그먼트화됩니다. 이 경우 GRE 터널 인터페이스에 **tunnel path-mtu-discovery** 명령을 구성하지 않습니다. 그 결과, IPv4sec 피어에서의 IPv4 패킷 재조합이 프로세스 스위칭 모드에서 수행되기 때문에 처리량이 크게 줄어들 수 있습니다.

관련 정보

- IP 라우팅 지원 페이지
- IPSec(IP Security Protocol) 지원 페이지
- RFC 1191 경로 MTU 검색
- RFC 1063 IP MTU 검색 옵션
- RFC 791 인터넷 프로토콜
- RFC 793 전송 제어 프로토콜
- RFC 879 TCP 최대 세그먼트 크기 및 관련 항목
- RFC 1701 GRE(Generic Routing Encapsulation)
- RFC 1241 인터넷 캡슐화 프로토콜 체계
- RFC 2003 IP 내 IP 캡슐화
- 기술 지원 및 문서 – Cisco Systems

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.